

2000
2000

2000/2000

2000/2000

2000/2000

PRODUCT DESCRIPTION

Protocol Tester K1197

SIEMENS AG

Revision Date : February, 1990

Module

UNITED

2010

2010

2010

PRODUCT DESCRIPTION

Product Description

2010

2010

2010

2010

2010

2010

2010

2010

2010

2010

2010

2010

2010

1	INTRODUCTION	1-1
2	ARCHITECTURE	2-1
2.1	GENERAL	2-1
2.2	HARDWARE	2-2
2.3	ENVIRONMENT CONDITIONS	2-4
3	DESCRIPTION OF THE APPLICATIONS	3-1
3.1	GENERAL DESCRIPTION	3-1
3.1.1	BUILDING BLOCK SYSTEM	3-1
3.1.2	MENU SYSTEM	3-2
3.2	CCS#7 SOFTWARE	3-3
3.2.1	CCS#7 MONITOR	3-4
3.2.2	CCS#7 SIMULATOR / EMULATOR	3-6
3.3	ISDN D-CHANNEL SOFTWARE	3-8
3.3.1	ISDN MONITOR	3-8
3.3.2	EMULATING / SIMULATING THE D-CHANNEL PROTOCOL	3-14
3.4	X.25 SOFTWARE	3-17
3.4.1	X.25 MONITOR	3-17
3.4.2	X.25 EMULATOR	3-22
3.4.3	X.25 TESTPACK EXTENSIONS	3-24
3.4.3.1	MULTILINK (MLP) MONITOR / SIMULATION	3-24
3.4.3.2	CCITT RED BOOK / YELLOW BOOK ADDITIONS	3-27



Digitized by the Internet Archive
in 2025

https://archive.org/details/manualsplus_05469

1 INTRODUCTION

This paper will give a comprehensive overview of most types, applications and features of the **PROTOCOL TESTER K1197**.

As the development goes on, each year brings optimized and additional applications.

The PROTOCOL TESTER K1197 is a test instrument to analyze, simulate and emulate data streams on lines with interfaces as described in CCITT Recommendations V. and X., ISDN Primary Rate, ISDN Basic Rate and others.

The instrument handles OSI Layers 1-7 in real time.

SIEMENS has developed the K1197 to satisfy the needs of the data communications industry for a powerful, yet easy to use protocol test machine. It combines the power and flexibility of a 4-th generation super micro computer with the economics and ease of use of a job oriented, menu driven test tool. The K1197 is equally suited for **field testing** or **development laboratory** where it will monitor or simulate most industry standard or custom interfaces and protocols.

The K1197 is packaged in a rugged, portable case which meets the weight and size requirements for airline carry-on baggage.

Design and architecture of the K1197 have been structured to ensure full upward compatibility with ongoing and future HW and SW-developments to meet not only the immediate but also the long term testing needs.

This brochure provides an overview of the functions and capabilities of the K1197 and its optional Test Packs. A Test Pack provides complete protocol testing functionality including:

- Monitor
- Protocol Simulation (Emulation)
- Test Manager
- Test Sequences

K1197 Dual Interface Protocol Tester

The K1197 interfaces

o ISDN

ISDN-BAI

Basic Access Interfaces:

S₀

U_{p0}

o ISDN and CCS#7

ISDN-PRI

Primary Rate Interfaces:

T1 for 1.544 MHz
CEPT for 2.048 MHz (PCM30)

o V/X

V/X-Interfaces:

V.28

V.11

V.36

V.35

two of the above interfaces
may be combined

The K1197 application packets

o ISDN:

D-Channel:
(Layer 2,
Layer 3)

- Monitor
- Emulator
- Simulator

for CCITT-88,
1TR6,
ETSI T/S 46,
and others

B-Channel:

- Monitor
- Emulator
- Simulator

for X.25,
T.90 and
TELEMATIK

o CCS#7:

CCS#7:

- Monitor
- Emulator
- Simulator

for TUP,
TUP+,
ISUP,
SCCP,
TF
and others

o V/X

- Monitor
- Emulator
- Simulator

for X.25,
X.75

two software packets can run independently
and simultaneously

To get more information about K1197 hardware and software moduls, call Siemens Reference Service, Customer Service at (0 89) 41 11-24 39.

Siemens AG, AUT V331, Communications Test Equipment, P.O. Box 70 00 79,
D-8000 Munich 70, West Germany.

2 ARCHITECTURE

2.1 GENERAL

The K1197's advanced capabilities are based on a unique system architecture which includes the following components:

- | | | |
|------------------------|-----|--|
| Hardware | : - | <ul style="list-style-type: none"> 32-bit multiprocessor system using 3 up to 7 MC 68000 (16 MHz) 3 - 9 Mbytes of fast dynamic RAM (expandable) Industry standard VME bus High resolution 8 color 9" CRT display Detachable full ASCII keyboard with 8 programmable function keys Standard 40 Mbytes harddisk drive (access time < 28 ms, auto retract) Dual floppy disk drives 3,5" (1,3 Mbyte/790 Kbytes formatted each) ISDN Primary Rate Interface (PRI) for T1 (1.544 Mbit/s) and CEPT (2.048 Mbit/s) ISDN Basic Rate Interface (BAI) for S-bus (192 kbit/s) V/X Interface (VXI), software selectable interface type Connectormodules for T1 (1.544 Mbit/s), CEPT (2.048 Mbit/s), S₀ (192 kbit/s), U_{p0} and V/X |
| System Software | : - | <p>The MF 2.X is a real-time, event driven, multitasking, multiprocessing operating system. It provides scheduling, interprocess communication, terminal console and disk I/O functions, and communications front ends.</p> |
| Utilities | : - | <p>The basic K1197 includes menu driven disk file management programs, a function key controlled full screen source editor and a printer I/O function</p> |
| Programming: | - | <p>The K1197 supports an enhanced version of FORTH. FORTH is a high level programming language and a development environment.</p> <p>FORTH supports the principles of modern programming style such as structured and object oriented programming, top down design, modularity, readability, minimized maintenance effort, and it produces fast and compact code. There are one or more concurrent FORTH machines.</p> |
| Applications : | - | <p>For advanced protocol testing, including fully automatic protocol emulations, Conformance tests and network performance measurements, Siemens offers optional TEST PAKS for X.21, X.25, T.90, X.75, TELETEX, ISDN D-Channel and B-Channel applications, CCS#7 and custom testing applications.</p> |

2.2 HARDWARE

HOUSING	Modular construction, polycarbonate injection molded housing with aluminium side reinforcements and aluminium back. Retractable handle extends from the side. Structures are mechanically tested to withstand shock and vibration normally expected from transportation and handling.
CRT	Built in high resolution 8 color CRT display 9 diag. 23 x 80 10 x 12 per character.
KEYBOARD	Detachable full ASCII keyboard with 8 programmable function keys.
DUAL 3.5" FLOPPY DISK DRIVES	DS, HD/DD 135 TPI. Total formatted storage capability per drive is 1,3 Mbyte/790 Kbytes.
3.5" HARDDISK DRIVE	Total formatted storage capability is 40 Mbytes. Average access time is 28 ms.
AUXILIARY PORTS	Asynchronous ports, 25 pin, V.28/RS-232C. ISO 2110
To MODEM	Remote port -to DCE configuration (male socket)
To PRINTER	Printer port -to DTE configuration (female socket)
EXT.CRT	RGB-TTL -8 pin DIN 45326 female connector
VME BUS	Standard configuration, 23 address lines
POWER SUPPLY	5 V / 30 A, 12 V / 3 A, -12 V / 0,8 A, 24 V / 2 A
BACKPLANE	7 slots for double eurocard size. It provides integral data and power connections to MAIN CPU, FECs, interface boards, CRT, disk drives, fan, and power supply.
SLOT 1 MAIN CPU	32 bit MC68000 CPU, 1 Mbyte of RAM CRT controller HD/FD controller Real-time calendar clock (battery backed up, 2 years) 3 16 bit counter/timer Dual serial asynchronous I/O ports
SLOT 2	Expansion slot
SLOT 3	Expansion slot
SLOT 4 FECP 1	Front end processor module for 1 full duplex test port : Two 32 bit MC68000 CPU's 2 Mbyte of RAM 80 I/O-Lines 4 HW timers, 1 ms resolution

SLOT 5 FEC P 2

Front end processor module for 1 full duplex test port :
Two 32 bit MC68000 CPU's
2 Mbyte of RAM
80 I/O-Lines
4 HW timers, 1 ms resolution

SLOT 6 INTERFACE 1

a) PRI :
T1 (23 B + D, 1.544 Mbit/s)
CEPT (31 B + D, 2.048 Mbit/s)

b) BAI :
S₀ (2 B + D, 192 kbit/s)
U_{p0} (2 B + D, 384 kbit/s)

c) VXI :
V.24
V.36
X.21
V.35

SLOT 7 INTERFACE 2

see SLOT 6

SELF TEST

Extensive power-up self diagnostic programs are incorporated into the system software.

2.3 ENVIRONMENT CONDITIONS

POWER REQUIREMENTS

VOLTAGE	230 V (115 V) +/- 15%, 47-63 Hz single phase. 500 VA max.
FUSE	3,15 A T fuse

WEIGHT

NET WEIGHT	Approximately 19,8 Kg (44 lbs.)
SHIPPING WEIGHT	Approximately 25 Kg (55 lbs.)

TEMPERATURE

OPERATING	+5 C to +40 C (40 F to 104 F) non-condensing
NON-OPERATING	-20 C to +60 C (-4 F to 140 F)

HUMIDITY

OPERATING	up to 80% humidity at +40 C (104 F)
NON-OPERATING	up to 95% humidity at +60 C (140 F)

ALTITUDE

OPERATING	up to 3,000 meters (10,000 ft)
NON-OPERATING	up to 12,000 meters (40,000 ft)

SHOCK

OPERATING	up to 2 G's
-----------	-------------

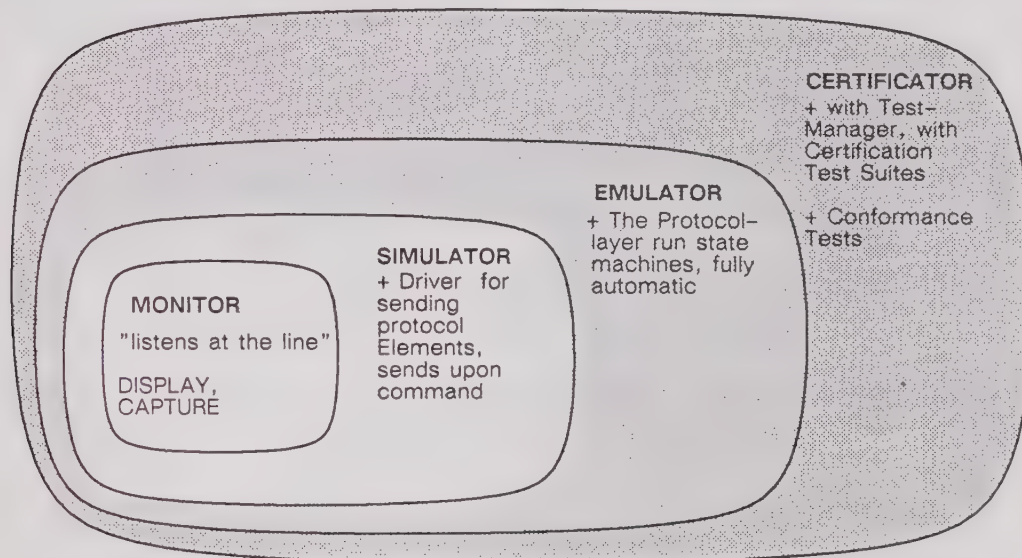
SAFETY STANDARDS

VDE 0411 (power supply, monitor X-ray radiation)
VDE 0860 (monitor tube implosion)
VDE 0871 Level B, German Bundespost Vfg. 483/86 and 1046/84

3 DESCRIPTION OF THE APPLICATIONS

3.1 GENERAL DESCRIPTION

3.1.1 BUILDING BLOCK SYSTEM



3.1.2 MENU SYSTEM

The menu system provides an intuitive interface designed to minimize the need to refer to the manual. The same system is used through out all applications. It includes pop-up help screens.

Example: **Menu selection: data playback**

```

SIM DRO 1A:ACTIVE 1B:NO SIGNAL 2A:ACTIVE 2B:ACTIVE 89-11-04 09:10:22
MONITOR LIVE DATA RECORD OFF CAPTURE Wrap 0064K free
                                L4M  SCCP  L3M  L2SF  TYPE  Dir  Blockno

+-----+
| REPORT MODES      >|
| TRIGGER / FILTER>| +-----+
| *DISK FUNCTION    >|--| DATA RECORDING >| +-----+
|                   | | DRIVE:      DRO      >>| |
| TEST MANAGER      >| | *DATA PLAYBACK >|--|
| PRINT             | | FILENAME: OUT_IN  |
| SETUP             >| | CAPTURE TO DISK>|
| UTILITIES         >| +-----+ EXECUTE |
+-----+                               +-----+

OK-1
1 QUIT MENU 2 HELP 3 COMMAND 4 5 6 7 8

```

The first three lines indicate the status of the machine.

The next lines display data or contain the menu structure.

The last line shows the soft keys.

3.2 CCS#7 SOFTWARE

The Siemens K1197 Common Channel Signalling System No.7 (CCS#7) Protocol Tester conforms to the CCITT Recommendations Q.700 through Q.704 and Q.721 through Q.725 with variants available for the Red Book (1984) and Blue Book (1988) editions of these recommendations. This release supports Analyzers and Simulators for Levels 1, 2, 3, the Telephone User Part (TUP), the Signalling Connection Control Part (SCCP) and the ISDN User Part (ISUP). It also supports Level 1, Level 2 and Level 3 Emulation on up to four signalling links.

The CCS#7 Tester supports:

- | | | |
|-----------------------|---|---|
| Level 1 (Q.702) | - | at the 1.5 and 2 Mbit/s interfaces, as well as speeds up to 64 kbit/s via the V.24/V.28, V.35, V.36, V.11 and X.21 interfaces |
| Level 2 (Q.703) | - | including the PCR and Double Transmission modes of error correction |
| Level 3 (Q.704) | | |
| Level 4 (Q.721-Q.724) | - | The Telephone User Part. |
| (Q.761-Q.764) | - | The ISDN User Part. |
| (Q.711-Q.714) | - | The Signalling Connection Control Part. |

National variants for different Signalling Point Code field sizes, different message types, different field types, etc. have been allowed for in the design. Several variants are available with this release. Others can be created if necessary.

The software packages contain either a monitor or a monitor/simulator/emulator combination. The special features are :

- Complete, fast decoding and display
(most comprehensive and complete decoding on the market, wide variety of user parts and display modes, including full mnemo display)
- Object oriented test language and CCITT SDL programmable state machine several user parts simultaneously available. They are easily interchangeable.
- Convenient user interface with window-oriented menu system. Details appear on screen only when needed.
- Message building system for easy handling of most complicated data types.
- Examples for most types of test suites/state machines are supplied – as basis for customized versions.
- Traffic densities up to nearly one erlang (= 100% line utilisation).
- Software packages support both PCM standards (CEPT or US) and V.24/V.11/V.35/V.36 interfaces.

3.2.1 CCS#7 MONITOR

Multi Link Monitor and Monitor/Simulator packages

Software package	User Parts
Level 2	--
TUP TUP+	CCITT Yellow, Red, Blue Book, CEPT incl. MTP
ISUP/SCCP CCITT	Red, Blue Book, DBP FTZ 1TR7 incl. TF (incl. MTP)

All of the CCS#7 Software packages contain a Monitor part in any configuration. The basic concepts are the same for all levels.

The CCS#7 Monitor provides for the comprehensive display of CCS#7 signalling traffic at level 2 to 4 in various display modes. Signalling traffic might be written to disk (recording) and/or stored in Capture RAM (capturing). *Filters* determine which signal units are displayed, written to disk or stored in Capture RAM. *Triggers* may be used to execute any action upon receipt of a specified signal unit.

Signalling information from both directions of a signalling link and from different signalling links is displayed simultaneously at the screen. Each link and direction is identified by a different color and label.

The information stored in the Capture RAM can be reviewed on the screen using any of the many different display modes. It can also be transferred to disk or to the printer if desired.

The information stored on a disk can be redisplayed or printed.

REPORT GENERATOR

The Report Generator module receives events (signal units, level 1 indications, trace messages) from the Report Filter and creates a report of this event based on user selected parameters. At Levels 2, 3 and 4, the user may specify a Short Report, various Medium Reports, a Complete Report, a Hex Report or a Character Report. The character set used in the Character Report may be set to ASCII or EBCDIC or HEX.

The report format for physical level (L1), link level (L2), level 3 (L3), SCCP and level 4 (L4) can be selected individually. The type of Report may be selected using the REPORT MODES menu or via the associated commands entered from the keyboard.

For referencing the frames, block number or timestamp display may be selected. For timestamp there exist three display types: year, hour, second.

Here is an example for screen display and printout :

SIM		DRO	1A:ACTIVE	1B:NO SIGNAL	2A:ACTIVE	2B:ACTIVE	89-11-04	09:10:22							
MONITOR		LIVE DATA		RECORD OFF		CAPTURE Wrap		0064K free							
		L4M	L3M	L2SF	TYPE	Dir	Blockno								
					SIO	LSSU	1A	1							
					SIOS	LSSU	1B	21045							
						FISU	1A	16550							
						FISU	1B	21258							
			STM			T&M	1B	22512							
						FISU	1B	22513							
						FISU	1A	17789							
			STA			T&M	1A	17796							
						FISU	1A	17797							
						FISU	1B	22525							
						FISU	1A	18327							
			STM			T&M	1A	33649							
						FISU	1A	33650							
						FISU	1B	38495							
			STA			T&M	1B	38503							
						FISU	1B	38504							
						FISU	1A	33662							
>															
OK-1															
1	QUIT MENU	2	HELP	3	COMMAND	4	HALT	5	FREEZE	6	MONITOR	7		8	TM KEYS

PROGRAMMABLE FILTERS AND TRIGGER

The CCS#7 Monitor incorporates programmable filters which allow the user to pass selected events and to filter out others. Only relevant information is displayed on the screen and unwanted information, such as FISUs, is not stored in the Capture RAM or on the Disk. The ability to filter the data received according to a specific source and destination is also provided.

TEST MANAGER

The CCS#7 Test Manager is used to execute a user-defined test scenario which may test the protocol and/or service conformance of an IUT (Implementation Under Test), or simply store or display the desired information about the monitored data. The Test Manager is included in Monitor and Simulator/Emulator software packets. Typical information that might be stored and/or displayed includes throughput performance data and the occurrence of a particular frame sequence. In order to set up a test scenario, a set of FORTH words is provided which enable the test to be defined in terms of a finite state machine.

The Test Manager can be invoked both by the Monitor and the Emulator. When it is invoked by the Monitor, it serves as an analytical tool which allows the user to collect specific data using a specified format on the desired device. When it is invoked by the Emulator, it not only has all the analytical power, but also the capability to send tailored frames to the test suite so that both protocol and service conformance of the test suite can be examined.

The Test Manager scenarios are organized into state machines to allow for clearly defined test procedures. Any event that is detected will cause execution of the code defined in the current STATE{...}STATE part. Then, if the event matches one of the events handled by the current "STATE" section, the corresponding ACTION{...}ACTION code is executed and entry to a new state may be defined if desired. Entry to a new state in this context means loading a new current state, which is executed after the **next** event is detected.

The state machines are programmed in a language similar to CCITT SDL Z.101-104.

3.2.2 CCS#7 SIMULATOR / EMULATOR

The software contains an emulation of MTP Level 2 / Level 3 on up to four links and simulation capabilities for various Telephony or ISDN User Parts.

Simulating User Parts or MTP Level 3 is based on the Test manger. Provision has been made to send messages in response to received messages, to operator commands or timeouts.

A message pool is used to store messages or message templates. Messages in the pool are identified by a unique name. They can be edited using the menu driven message building system, the hex editor or specifying the message contents using the FORTH oriented method of specifying a message. Strings (containing a message) can be stored in the pool directly. Message pools can be saved to disk files and can be reloaded and merged from disk files.

CCS#7 MTP EMULATOR

The emulation of MTP Level 2 contains the Basic Transmission Control and sending and responding to Signalling Link Test Messages at Level 3. It may be switched off.

Up to four separate MTP level 2 emulations (one for each signalling link) are available in the K1197.

CCS#7 USER PART SIMULATOR

The CCS#7 User Part Simulator allows to issue e.g. send commands or to perform user specified test sequences to be performed on CCS#7 links. The software contains a revised set of simulation commands suitable for the generation of Level 2, Level 3, SCCP and Level 4 test sequences.

MESSAGE BUILDING SYSTEM

The Message Building System (MBS) allows all legal and some illegal messages to be built by editing all the component fields in a mode reminiscent of the complete mode report in the monitor section of the package. The MBS takes care of all the details of the coding of messages and allows the user to concentrate on the data within the fields. It allows the values of bit fields to be set by selecting the appropriate mnemonic string from all the available options rather than having to look up the correct value in the standard. The MBS is entirely menu driven and operates on messages stored in the message pool.

Here an example, using the menu system :

Menu selection : simulation, message pool, IAM, edit, mnemonic, telephone user part, IAM

```

+---+-----Initial Address Message-----+
| R| ----0100 Service Indicator       : Telephone User Part      >>|
| T| 1000---- Sub-Service Field       : National message       >>|
| D| ***** Destination Point Code   : 10091                  |
| S| ***** Originating Point Code   : 10081                  |
| T| ***** Circuit Ident Code       : 33                      |
| P| ----0001 Heading code 0          : 1                      |
| S| 0001---- Heading code 1          : 1                      |
| U| --010000 Service Category        : Sub with CLI          >>|
+---| 00----- Spare                 :                          >>|
    | -----01 Nature of address indic : Not Identified        >>|
    | ----00-- Satellite indicator      : No satellite Link     >>|
    | --00---- Continuity check indicat : Continuity check not required>>|
    | -0----- Echo Control Device indic : Half Echo Cont Device not incl>|
    | 0----- Spare                   :                          >>|
    | -----0 Redirected call indicator : Not a redirected call  >>|
+---+-----+

```


3.3 ISDN D-CHANNEL SOFTWARE

The ISDN D-channel software contains a monitor, a test manager, a full emulation of a terminal equipment or a network terminator up to layer 2 and a simulator up to layer 3 according to CCITT recommendation Q.920, Q.921, Q.930 and Q.931. Several national variants for layer 3 are available, such as 1TR6 of the Deutsche Bundespost and T/S 46-30 of the ETSI.

The software is controlled by a menu system which is designed to minimize the need to refer to the manual.

3.3.1 ISDN MONITOR

All of the ISDN software packages contain a monitor part in any configuration.

The ISDN monitor provides a real-time trace of the line traffic and interface condition. Various display modes are available for the comprehensive display of ISDN D-channel signalling traffic at layer 1 to 3. These monitor formats operate independently of each other so that the user can have any combination of display formats desired.

Signalling traffic might be written to disk (recording) and/or stored in Capture RAM (capturing). *Filters* determine which information is displayed, written to disk or stored in Capture RAM. *Triggers* may be used to execute any action upon receipt of a specified information.

Signalling information from both directions of the line traffic is displayed simultaneously at the screen. Each direction is identified by a different color and label.

The information stored in the Capture RAM can be reviewed on the screen using any of the many different display modes. It can also be transferred to disk or to the printer if desired.

The information stored on a disk can be redisplayed or printed.

Data Flow Diagram

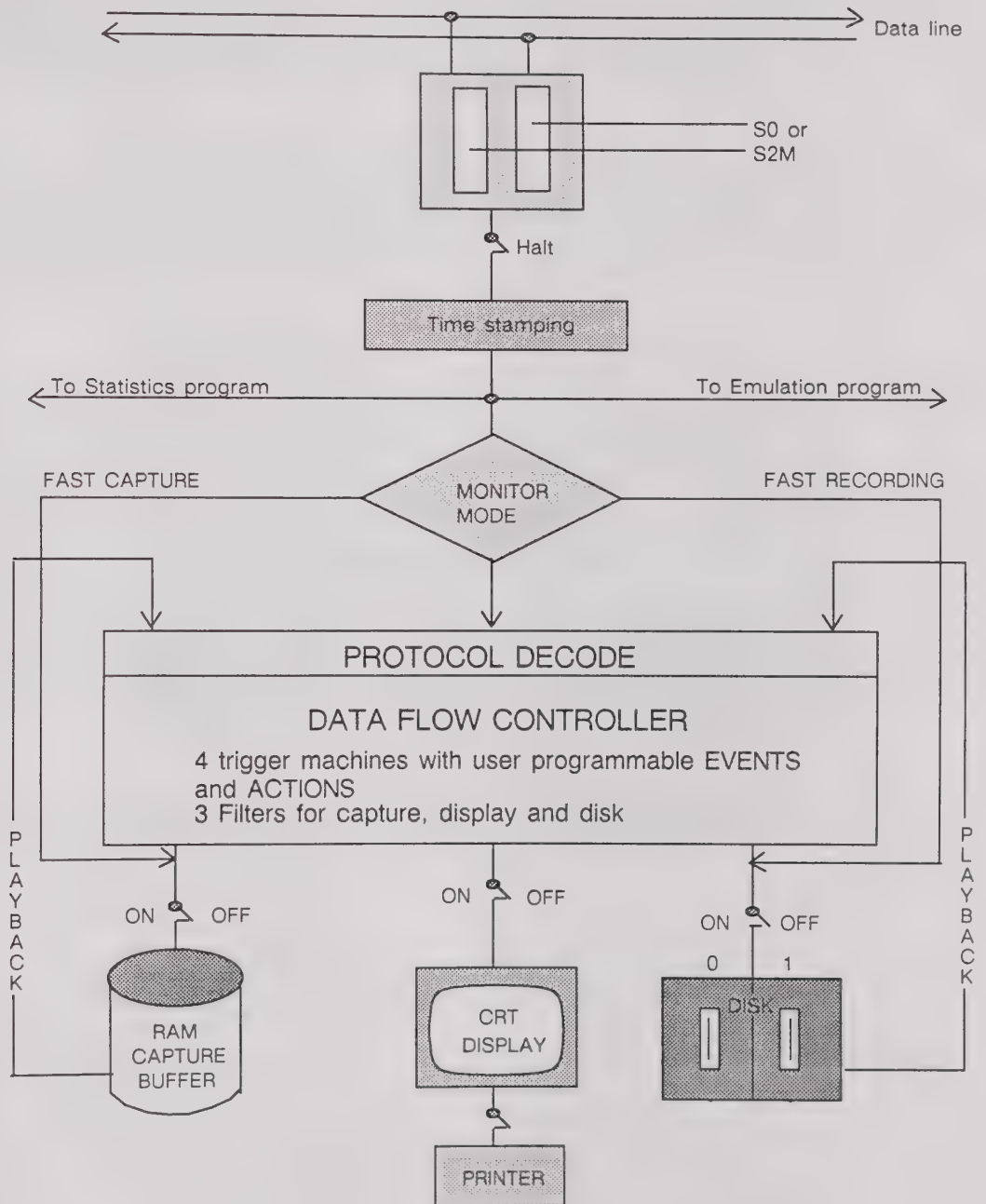


Figure 3.3.1-1: Data Flow Diagram

Protocol Decode

This module decodes the data received from the line or playback media and determines the frame or packet type and other protocol header information. The decoded information is then passed to either the data flow controller to check for active triggers and filters or the Test Manager for statistics calculations, performance measurements or the evaluation of protocol violations.

Example for a complete LAP-D decoding and display

BLOCK_NUMBER	ORIG	SAPI	TEI	C/R	FRAME	P/F	N(R)	N(S)
1	Net	63	127	1	UI	P=0		
					Message name			: Identity assigned
					Management entity identifier			: 15
					Reference number (RI)			: 7752
					Message type			: 2
					Action identifier (AI)			: 64
0	Usr							
3	Usr	0	64	0	SABME	P=1		

Example for a complete message decoding and display

BLOCK_NUMBER	ORIG
2	Usr
Protocol Discriminator = Q.931 Setup	
Call reference = 4660 Orig	
1	00000100 INFORMATION ELEMENT : Bearer capability
2	00000011 IE Length : 3 Octets
3	1----- Extension Bit : Not Continued
	-00----- Coding Standard : CCITT
	---00000 Info Trans Cap : Speech
4	1----- Extension Bit : Not Continued
	-00----- Transfer Mode : Circuit Mode
	---10000 Transfer Rate : 64 kbits/s
5	1----- Extension Bit : Not Continued
	-01----- Layer Identification : User Info Layer 1 Protocol

Data Flow Controller

This SW module consists of 4 programmable trigger machines and 3 independent filter mechanisms. The trigger machines can be set up to start and stop the data flow and to time and count events. The filter allow the user to specify and capture, display or record selected data.

Programmable Triggers

Up to 4 trigger machines can be activated simultaneously. Each trigger machine can consist of multiple events and actions. The trigger machines can be used to control (start and stop) the data flow to the display, the capture buffer and the disk or to count and time stamp specific events. The trigger events are selected from menus and the actions are entered from the keyboard.

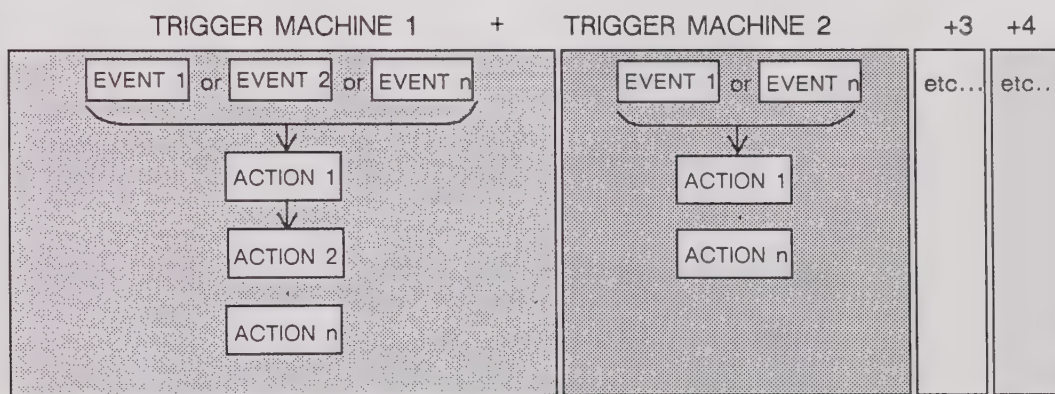


Figure 3.3.1-2: Different trigger machines in parallel

Trigger events are:

- the source of the data i.e. NTKW, USR, BOTH or PLAYBACK,
- the date and time,
- all layer 1 events,
- all frame types
- all message types
- specific LAPD and layer 3 addresses.
- received character strings.

Trigger actions are:

- start / stop display, capture and recording
- audible and visual alarm (highlighting),
- check condition flags, manipulate 128 counters
- activate/deactivate triggers,
- redefine trigger events and actions
- generate trace reports and messages
- event time stamp processing,
- most FORTH words.

Programmable Filters

The filters allow the user to select specific data or protocol data units for display, capture buffer storage and disk recording.

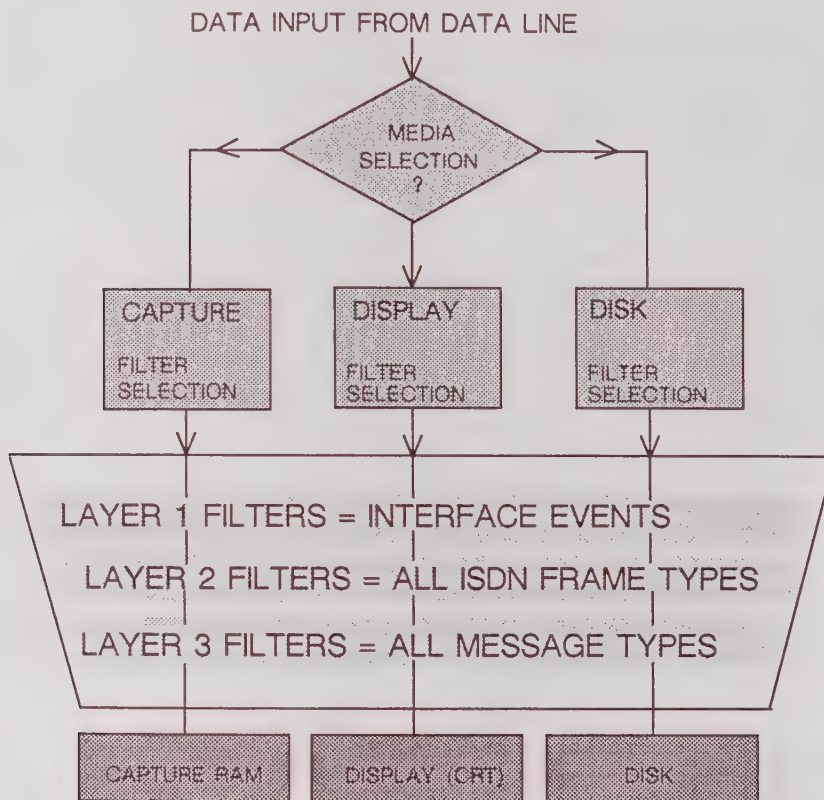


Figure 3.3.1-3: Data input to capture, display or disk

EXAMPLE:

A filter could be selected which only reports data on a particular logical channel to the display but at the same time another logical channel could be captured to RAM while, possibly all data is recorded to disk.

ISDN D-CHANNEL TEST MANAGER

The Test Manager, part of each software pack, is used to build and execute test sequence programs for Layer 2 and 3. Each test program can consist of up to 256 test states, each combining multiple events and actions.

The flexibility of the emulation programs has been further advanced by the use of the Test Manager for creation of simple customized test sequences or complex certification tests. The user interface to the Test Manager is via high level commands for creation and execution of all test sequences which are built on the K1197 in an off-line mode and then stored to disk in a test file. These files can then be called up and executed during an emulation of a TE or NT. The Test Manager is controlled via function key or user commands from the keyboard.

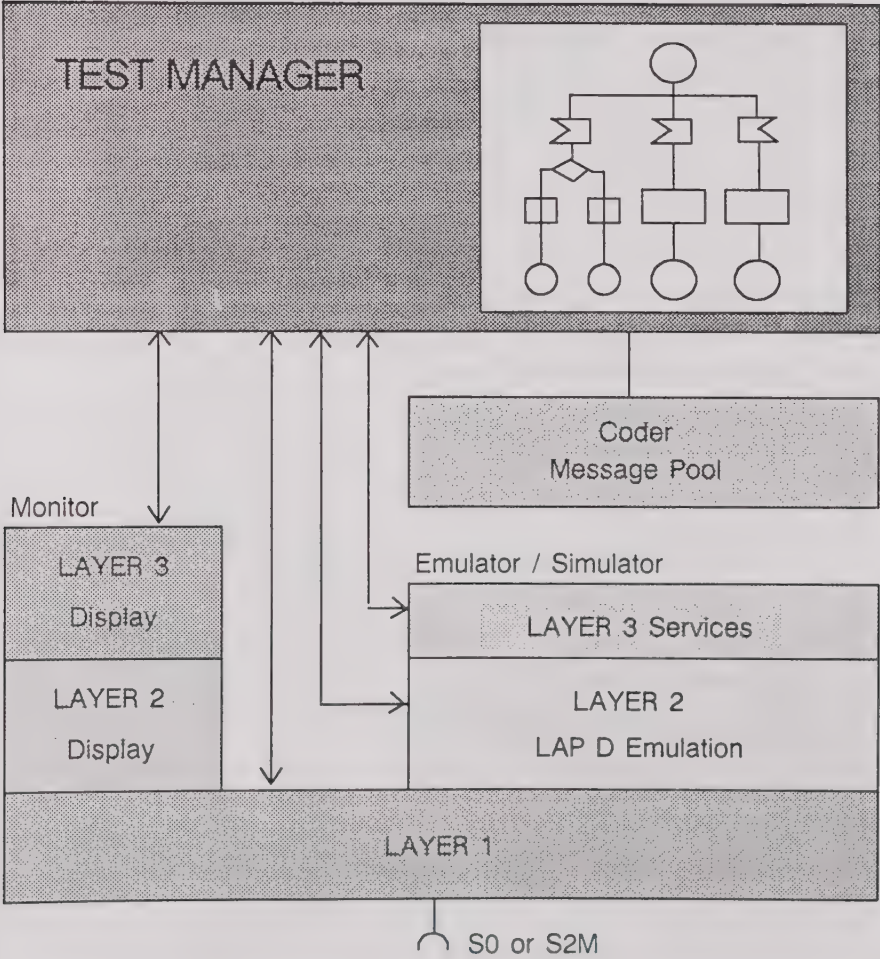


Figure 3.3.1-4: Interworking between Testmanager, Monitor and Emulator / Simulator

3.3.2 EMULATING / SIMULATING THE D-CHANNEL PROTOCOL

The ISDN emulation/simulation program can be configured to function as a user (TE) or network (NT) emulation and supports the following functions/features:

LAYER 2 (Q.921) I.441 SUPPORTED FEATURES:

1. MULTIPLE FRAME ESTABLISH/RELEASE PROCEDURES
2. MULTIPLE FRAME ACKNOWLEDGED OPERATION
 - MODULO 128
3. SINGLE FRAME UNACKNOWLEDGED OPERATION
4. TIMER RECOVERY PROCEDURES
5. NETWORK AND USER EMULATION
6. MULTIPLE TERMINAL (8) OPERATION
7. TEI (TERMINAL ENDPOINT IDENTIFIER) ASSIGNMENT
 AND CHECK PROCEDURES
- 8.3 SAPI's (SERVICE ACCESS POINT IDENTIFIERS)
 - SAPI 0 - SIGNALLING FRAME
 - SAPI 16 - PACKET DATA
 - SAPI 63 - MANAGEMENT FRAME

Figure 3.3.2-1: LAYER 2 (Q.921)

**LAYER 3 (Q.931) I.451 FEATURES
AVAILABLE FOR SIMULATION:**

1. CIRCUIT SWITCHED CALL PROCEDURE
2. L3-MULTIPLEXER
 (CONTROL OF CALL REFERENZ / CES / PD)
3. SUPPLEMENTARY SERVICES
 (USER TO USER / DETACH / SUSPEND-RESUME /
 FACILITY)
4. L3-TIMER CONTROL
5. MESSAGE BUILDING SYSTEM (MENU)
 PROGRAMABLE MESSAGES
6. SERVICES TO LAYER 2

Figure 3.3.2-2: LAYER 3 (Q.931)

The LAP-D (layer 2) emulation program is designed as a finite state machine. The user can control the state machine by means of keyboard commands, menus and stored test sequences. Commands are provided to initiate the state machine, generate service primitives and PDU's and control the results display. The Test Manager can be programmed to simulate the layer 3 and higher layer functions.

Initialisation

- Forces the state machine into any desired situation.
- User control of:
 - initial states for layer 2;
 - initial sequence number and counter value;
 - window variables; USER/NETWORK and SAPI's.

Control of link layer variables

- Set/Reset Poll bit, Response bit and W, X, Y, Z bits in FRMR
- Set SAPI, TEI value, N(r)/N(s) counts.

Send frames

Keyboard entries for user generated frames and conditions:

- SABME, DISC, DM, FRMR, etc...,
- MAKE_BUSY, RESET_BUSY to toggle busy state and
- SEND_P to send user defined packets.

Send packets

The user can define:

- number of packets to be sent,
- the packet size,
- SAPI value,
- bit pattern and
- the data field content.

LAYER 3 to LAYER 2 primitives

Primitive generation menus include:

- ESTABLISH a layer 2 connection
- RELEASE a layer 2 connection
- SEND a layer 3 DATA packet (using acknowledged or unacknowledged operation),
- ASSIGN or REMOVE TEI value and TEI check procedures.

Message building system

Powerful menu driven tool to:

- Generate layer 3 messages in two different modes (automatic and manual)
- Select desired IEs, octets and field entries.
- Manipulate generated messages.
- Store messages in a pool to save and load on/from disk.
- Send messages.
- Programming method of generating messages within the testmanager.

3.4 X.25 SOFTWARE

3.4.1 X.25 MONITOR

The X.25 Monitor program is supplied as part of the X.25 test pack. This program combines the real-time monitoring functions of the standard K1197 monitor (USM) with additional protocol specific decode, trigger filter and display capabilities.

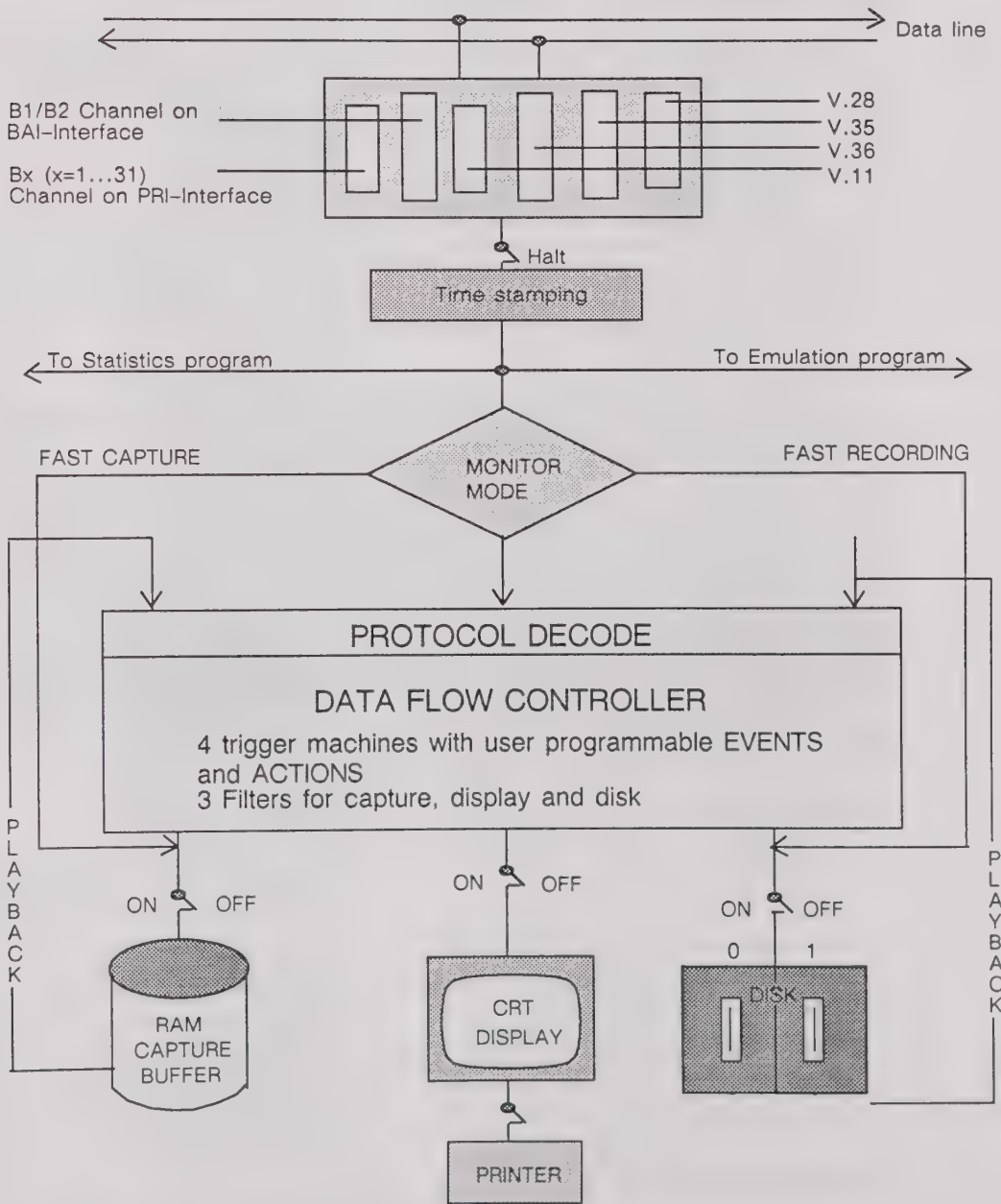


Figure 3.4.1-1: Data Flow Diagram

Protocol Decode

This module decodes the data received from the line or playback media and determines the frame or packet type and other protocol header information. The decoded information is then passed to either the data flow controller to check for active triggers and filters or the Test Manager for statistics calculations, performance measurements or the evaluation of protocol violations.

Data Flow Controller

This SW module consists of 4 programmable trigger machines and 3 independent filter mechanisms. The trigger machines can be set up to start and stop the data flow and to time and count events. The filter allow the user to specify and capture, display or record selected data.

Programmable Triggers

The 4 trigger machines can be activated in parallel. Each trigger machine can consist of multiple events and actions. The trigger machines can be used to control (start and stop) the data flow to the display, the capture buffer and the disk or to count and time stamp specific events. The trigger events are selected from menus and the actions are entered from the keyboard.

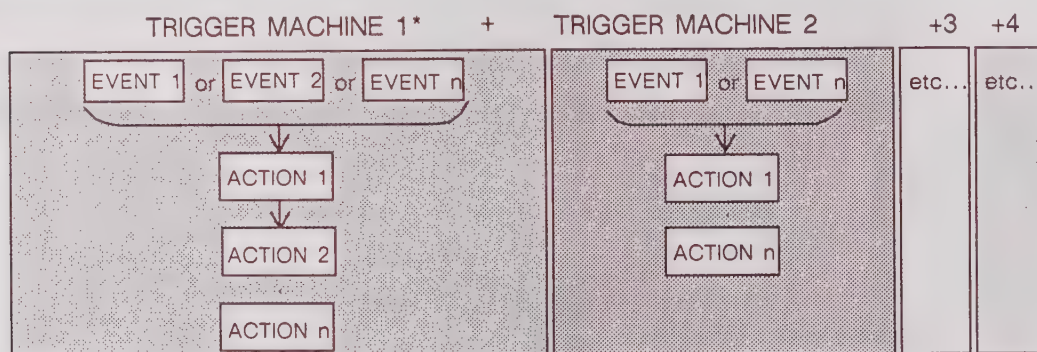


Figure 3.4.1-2: Different trigger machines in parallel

Trigger events are:

- the source of the data i.e. DCE, DTE, BOTH or PLAYBACK,
- the date and time,
- CRC errors, frame abort, short frames,
- interface control lead status or transitions
- all frame and packet types
- specific logical channels or network addresses.
- received character strings.

Trigger actions are:

- start / stop display, capture and recording
- audible and visual alarm (highlighting),
- check condition flags, manipulate 128 counters
- activate/deactivate triggers,
- redefine trigger events and actions
- generate trace reports and messages
- event time stamp processing,
- most FORTH words.

Programmable Filters

The filters consist of a set of high level user commands which allow the user to filter specific data or protocol data units to the display, the capture buffer and the disk.

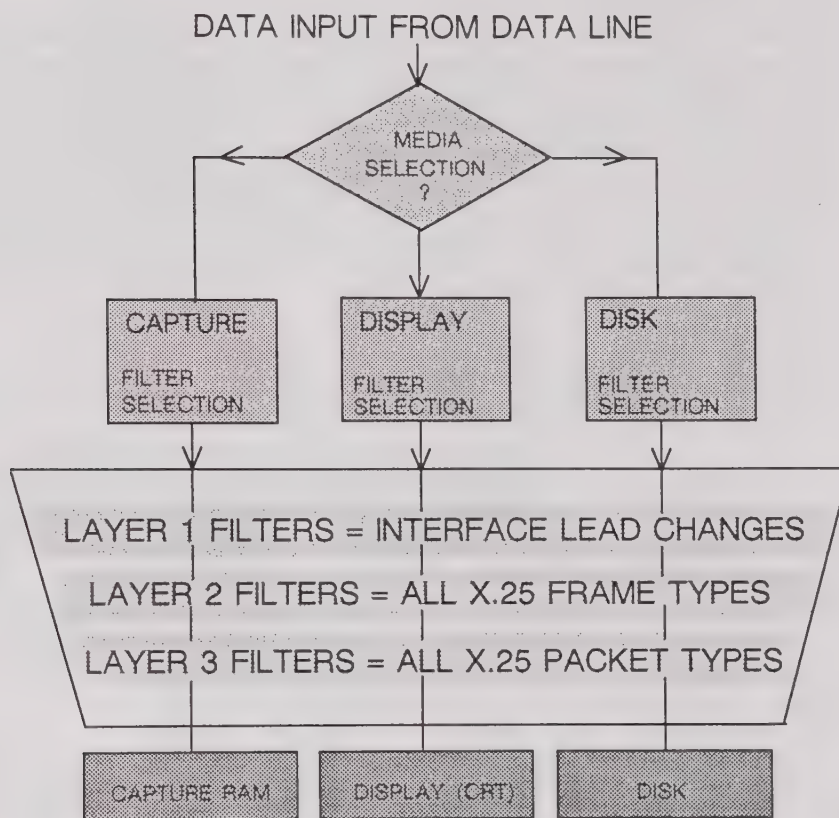


Figure 3.4.1-3: Data input to capture, display and disk

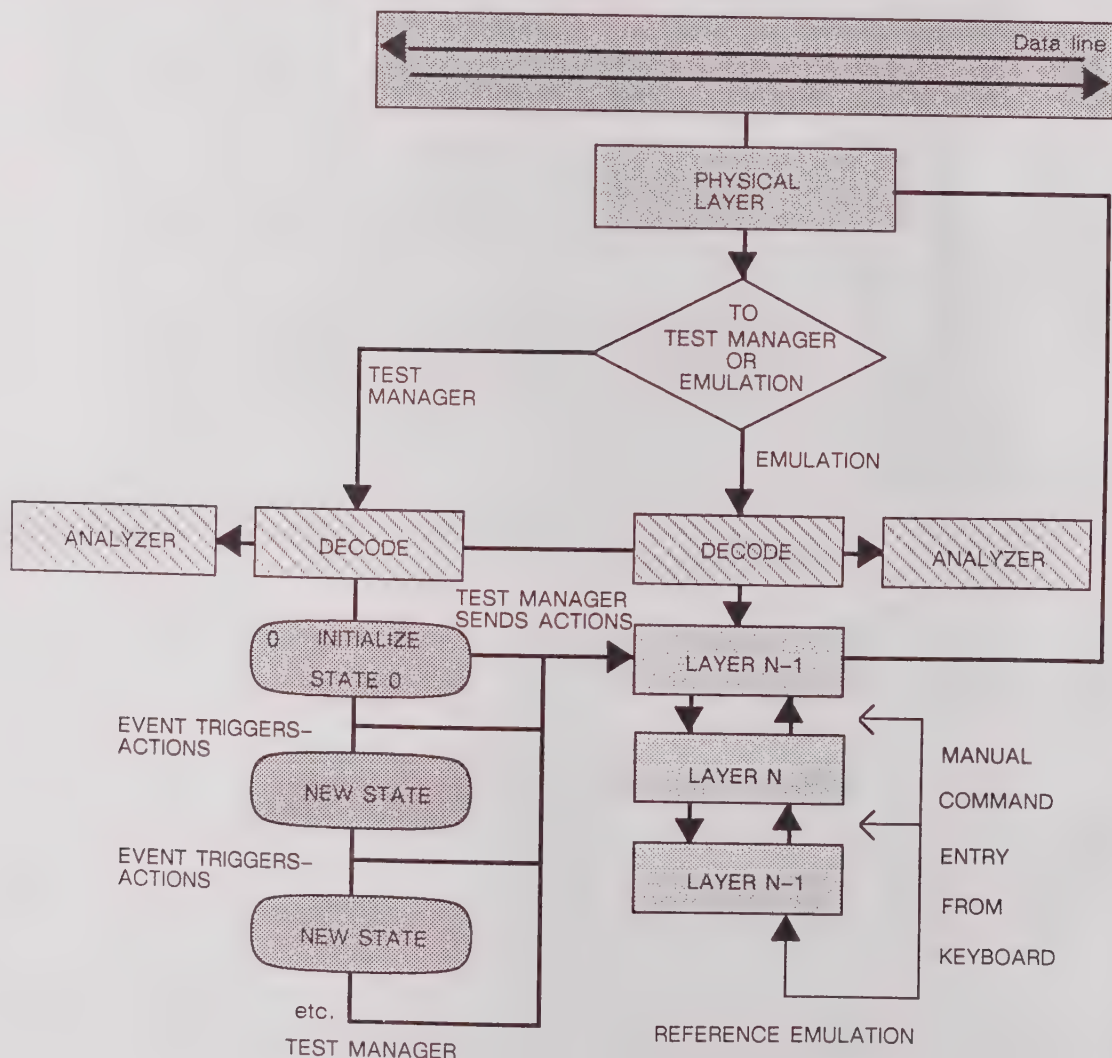
EXAMPLE:

A filter could be selected which only reports data on a particular logical channel to the display but at the same time another logical channel could be captured to RAM while, possibly all data is recorded to disk.

X.25 TEST MANAGER

The flexibility of the Monitor and Emulation programs has been further advanced by the use of the Test Manager for creation of simple customized test sequences or complex certification tests. The Test Manager is always included in monitor, simulator or emulator software packets. The user interface to the Test Manager is via high level commands for creation and execution of all test sequences which are built on the K1197 in an off-line mode and then stored to disk in a test file. These files can then be called up and executed during an emulation of a DTE or DCE. The Test Manager is controlled via 'soft' key or user commands from the keyboard.

The following diagram shows the interworking between the X.25 Reference Emulation, the Monitor and the Test Manager:



Data Display

There are four basic display modes: hexadecimal mode, complete mode, short mode and character mode.

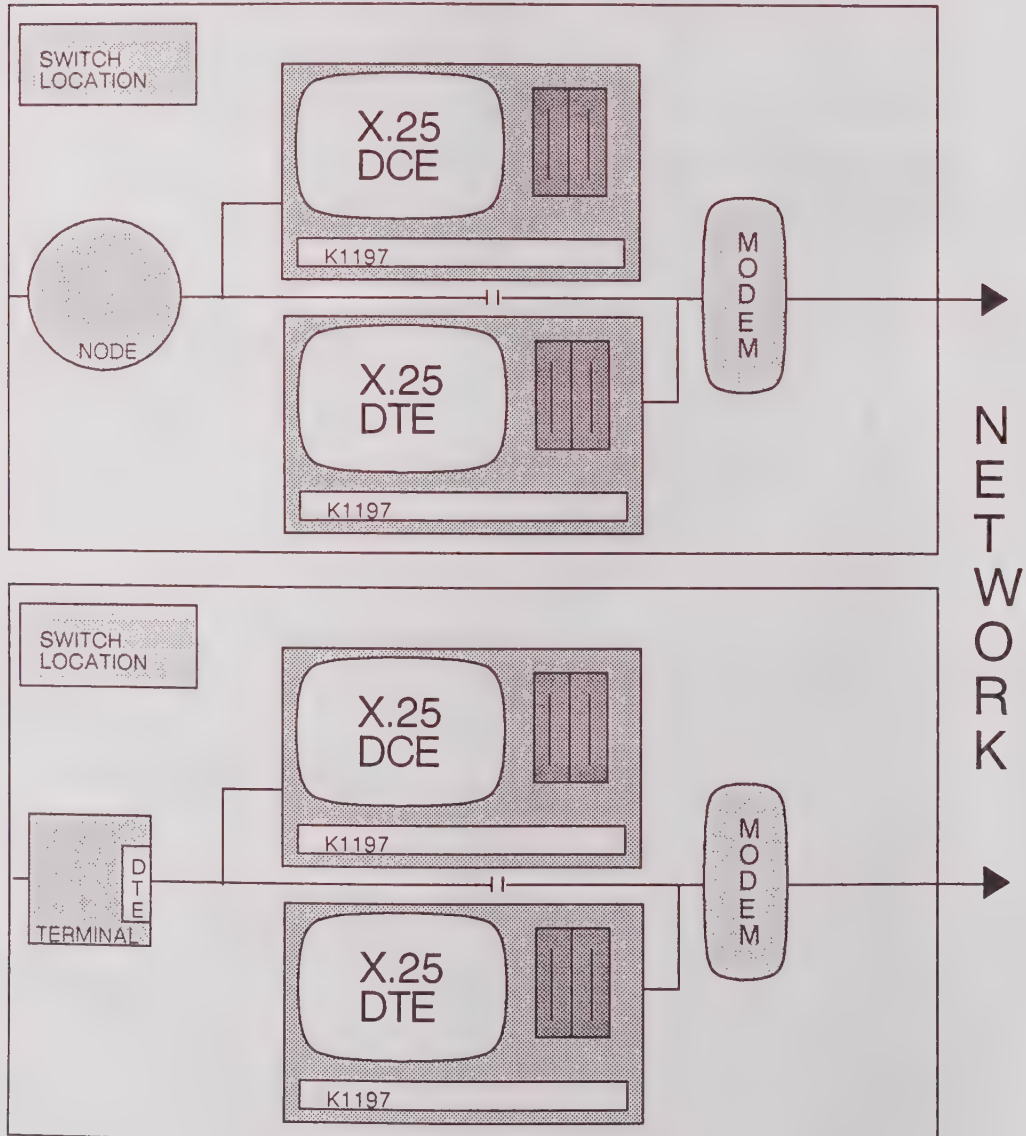
Example: Complete X.25 decoding and displaying:

TIMESTAMP	ORIG ADDRESS	COMPLETE FRAME AND PACKET REPORT			
21'843"405	DCE ADDRESS=01	FRAME=SABM	P=0		
21'860"077	DTE ADDRESS=01	FRAME=UA	F=0		
27'155"933	DTE ADDRESS=03	FRAME=INFO	P=0	NR=0	NS=0
	GF=1	LCN=0	RESTART INDICATION PACKET		
	CAUSE	= 00 UNDEFINED			
	DIAGNOSTIC	= 00 NO ADDITIONAL INFORMATION			
27'178"934	DCE ADDRESS=03	FRAME=RR	F=0	NR=1	
27'356"368	DCE ADDRESS=01	FRAME=INFO	P=0	NR=1	NS=0
	GF=1	LCN=0	RESTART CONFIRM PACKET		
27'374"922	DTE ADDRESS=01	FRAME=RR	F=0	NR=1	
29'605"762	DTE ADDRESS=03	FRAME=INFO	P=0	NR=1	NS=1
	GF=1 D=0	LCN=1	INCOMING CALL PACKET		
	ADDRESS	- CALLED = 36600011	CALLING = 45500011		
	FACILITY	- THROUGHPUT CLASS	- CALLED = 9600	CALLING = 9600	
	FACILITY	- SIZE	- CALLED 256	CALLING = 256	
	FACILITY	- WINDOW	- CALLED 3	CALLING = 3	
	CALL USER DATA FIELD				
	C ₀ 0 ₀ 0 ₀ 0 ₃ 1 ₀ 2 ₅ 8 ₀ 0 ₆ 4				
29'633"957	DCE ADDRESS=03	FRAME=RR	F=0	NR=2	
29'647"285	DCE ADDRESS=01	FRAME=INFO	P=0	NR=2	NS=1
	GF=1 D=0	LCN=1	CALL ACCEPTED PACKET		
	ADDRESS	- CALLED = 36600011	CALLING = 45500011		
	FACILITY	- THROUGHPUT CLASS	- CALLED = 9600	CALLING = 9600	
	FACILITY	- SIZE	- CALLED 256	CALLING = 256	
	FACILITY	- WINDOW	- CALLED 3	CALLING = 3	
29'679"829	DTE ADDRESS=01	FRAME=RR	F=0	NR=2	
31'984"347	DTE ADDRESS=03	FRAME=INFO	P=0	NR=2	NS=2
	GF=1 D=0 Q=0	LCN=1	DATA PACKET PR=0	PS=0	
	[SIEMENS PROUDLY PRESENTS YOU THE MULTIPOINT PROTOCOL TESTER]				
	[	THAT LEADS THE WAY INTO THE FUTURE OF DATA			]
	[	COMMUNICATIONS 128			]
32'112"204	DCE ADDRESS=03	FRAME=RR	F=0	NR=3	
32'122"199	DCE ADDRESS=01	FRAME=INFO	P=0	NR=3	NS=2
	GF=1 D=0 Q=0	LCN=1	RR	PACKET PR=1	

3.4.2 X.25 EMULATOR

This program has been designed for the emulation of X.25 and supports CCITT Rec.1980/84 and other relevant industry implementations. Like all other K1197 application programs it is provided on a 3.5 inch floppy diskette. Once loaded from disk the complete Emulation program is memory resident. The Emulation program is linked and compiled together with the X.25 Monitor program which provides the functions for data display, capture and recording. Also provided are a number of high-level (plain English) user commands for the configuration and control of the Emulation program.

Figure 3.4.2-1: The emulation supports the following configurations:



DCE plus DTE may be run Simultaneously on one K1197. Both of them may exchange Data via the test manager.

Program Configuration

The X.25 Emulation program provides commands to:

- select interface type (RS232, V.35, V.36, V.11)
- select DCE or DTE emulation
- select clock speed (2400 bps to 25600 bps)
- set frame parameters i.e. window size, retry counter, timer value
- set packet parameters i.e. window size, packet size
- assign network addresses for 64 logical channels
- define facility and user data fields
- define messages (10 buffers of 80 bytes and 1 of 1024 bytes)

Physical Layer Operation

- manual control of interface control leads

Frame Level Operation

- manual frame level mode
- automatic frame sequence numbering
- automatic frame level RR processing
- fully automatic frame level processing
- user defined frames (both legal of illegal)
- generate CRC errors

Packet Level Operation

- manual packet level mode
- automatic packet sequence numbering
- automatic packet level RR processing
- fully automatic packet level processing
- enable/disable facility negotiation
- generate user defined packets (both legal of illegal)

Operating Mode

- For Manual Testing the X.25 test pak allows the user to enter commands from the keyboard in order to manipulate the configuration parameters as well as generate data.
- These same commands, together with any other FORTH word, can be programmed into a test sequence for Automatic Testing with the help of the Test Manager.

Manual Testing

The actual commands which the user enters from the keyboard have the following formats:

SABM	: Send a SABM frame to establish the Link
RESTART	: Send a RESTART packet
CH27 CALL	: Send a CALL REQUEST packet on LCN27
DATA9	: Send a DATA packet (user defined data in buffer = 9)
CH27 CLEAR	: Send a CLEAR REQUEST packet on LCN27
DISC	: Send a DISC frame to disconnect the Link.

Test Manager

The function has been described in chapter 3.4.1 X.25 Monitor.

3.4.3 X.25 TESTPACK EXTENSIONS

3.4.3.1 MULTILINK (MLP) MONITOR/SIMULATION

This paper is a short form of the manual C73000-M6076-C187. It gives a description of the Multilink Procedure (CCITT Red Book 1984) and Extended Sequence Numbering modulo 128 features which were added to the X.25 Test Pack. They provide the ability to send and receive MLP PDU's in Emulation, Monitor and Test Manager. In addition, the X.25 Extended Sequence Numbering feature is supported in both the Link and Packet layers.

The new abilities can be activated or deactivated by simply switching them on or off. Also, because these new features of the package are independent of one another, any combination of them can be in use while the others are not. For example, both the MLP feature and the Link Layer Extended Sequence Numbering feature can be in use, while the Packet Layer uses basic (mod 8) sequence numbering.

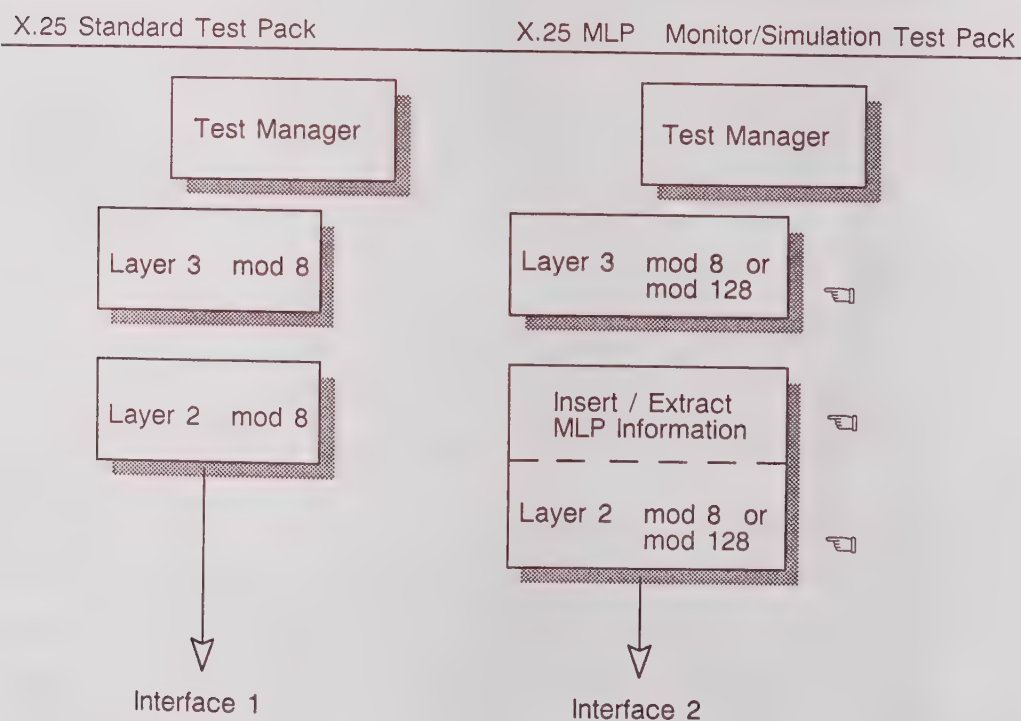


Figure 3.4.3.1-1: Differences between standard X.25 Test Pack and X.25 MLP Monitor / Simulator

The MLP simulation enables the user to send, receive and decode X.25 PDUs including the MLP layer. However, it does not provide an automatic full emulation of the MLP protocol. In fact, the package operates exactly as the standard X.25 package with the exception that an MLP PDU can be included in sent frames and extracted from received frames. Therefore, a single emulation (X.25 MLP package running on one CPU) can still control only one physical port at a time. It is on the other hand possible to achieve some form of dual port control by connecting two testmanager scenarios over two emulations.

The MLP simulation can be run in either SLP mode or MLP mode. In SLP mode, the X.25 Emulation and Monitor operate as normal with MLP PDUs neither sent nor expected in any frames. In MLP mode, any Packet layer packet sent (e.g. using the standard layer 3 send commands SENDP, SENDD, CALL, CLEAR, etc.) will have included an MLP PDU.

The different frame and packet layer formats are shown in the following picture:

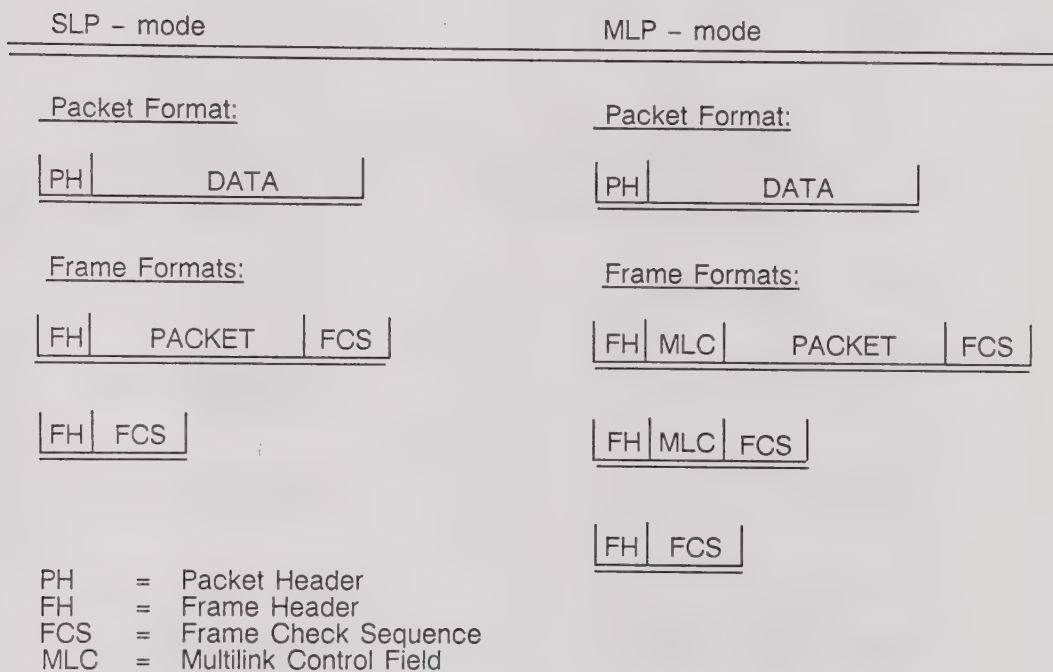


Figure 3.4.3.1-2: Packet and Frame formats

MLP Simulation on the TEST MANAGER

There have been three pairs of new FORTH words added to the Test Manager commands which are designed to coordinate two SLP emulations in order to simulate an MLP implementation. It is assumed that two emulations are running on one K1197 (one on each front end). Each emulation controls one link and the Test Manager scenarios – written for MLP testing and using the newly designed words – will coordinate the interaction of both Test Packs.

The scenarios for the two Test Managers are contained on floppy .

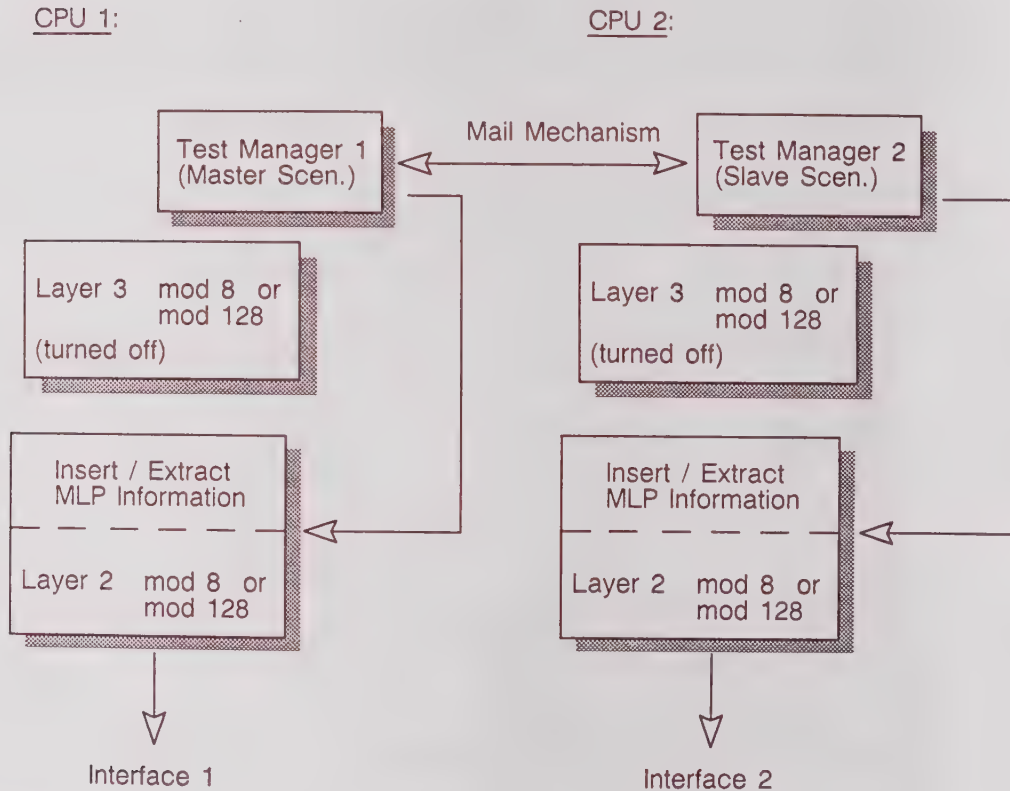


Figure 3.4.3.1-3: Principle of MLP Simulation (coordination of two SLP emulations via Test Manager)

MLP EMULATION

This testpak is in preparation

3.4.3.2 CCITT RED BOOK/YELLOW BOOK ADDITIONS

Red Book Chapter	Previous Test Pak	Changes
1.1. X.21 Interface with		Can be combined new X.21 Test Pak to allow full X.21 plus X.25
2.4.8. List of LAP B System Parameters		
2.4.8.3. Parameter T3	Forth : T1-IDLE	name changed to T3-Value
2.4.8.5. N1 max. # of bytes in I-Frame?	non existent .. FRAME_SIZE	name changed to SET_N1_S
3. Packet Level		
3.2. Basic structure of packets	Registration Request / Confirmation Emulation : Not supported Monitor : Display only in Hex	simulation added no emulation mnemos added
6.4. Packet retransmission L3 REJ	Monitor : supported Emulator : not supported	Emulation supported
Annex E Table E-1 Coding of diagnostic fields	Diagnostics 77, 96 not supported	added

Further additions :

Incompatible parts between red and yellow book, are selectable
by software switch.

- SABM Collision Handling
- Interrupt User Data Handling
- Receiving SABM, DISC with P=0

Separate parameter sets for N1 for both data directions.

Separate window size per logical channel.

Per-Call facility negotiation via table and setup file.

DBP facilities Sammelanschluss (Hunt Group) and Ruf-Weiterleitung (Call redirection).

Program: Operating System for PT K1197
Software Version: V2.2 Rev.1
Previous Version: V2.2 Rev.0
Revision Date: December, 1991

☞ Please read this information carefully before you work with the K1197 and the Operating Instructions. The modifications and extensions described here have a significant influence on the total functions of the K1197.

Installation:

Please exchange the files from disk with the winchester partition WD0.

After a reboot the new system is installed.

Changes Made:

1. The problems upon disk-recording made from any application to a full winchester partion are solved now.
2. There were extremly boot-problems which are solved now.
3. The DEL-key works correctly now.
4. It is not possible to make playback with non-data files anymore.
5. Some new applications are added to the menu selection.

USER MANUAL

Protocol Tester K1197

SIEMENS AG

System Version : MF2.2
Revision Date : February, 1991



WARNING

Hazardous voltages are present in this electrical equipment during operation. Non-observance of the safety instructions can result in severe personal injury or property damage.

Only qualified personnel should work on this equipment after becoming thoroughly familiar with all the warnings, safety notices and maintenance procedures contained in this documentation.

The successful and safe operation of this equipment is dependent on proper handling, installation, operation and maintenance.

RELEASE NOTES	R-1
1. GETTING TO KNOW YOUR K1197	1-1
1.1. FRONT AND BACK VIEWS	1-1
1.2. SIDE VIEW AND KEYBOARD	1-2
2. PREPARING FOR OPERATION	2-1
2.1. WARNING	2-1
2.2. FOR THE FIRST TIME USER	2-2
2.3. OPERATING CONDITIONS	2-3
3. GETTING INTO OPERATION	3-1
3.1. POWER ON & RESET DIAGNOSTICS	3-1
3.1.1. DISPLAY	3-1
3.1.2. DIAGNOSTIC TEST	3-2
3.1.3. FLOW CHART	3-4
3.1.4. FAILURE ON DIAGNOSTICS	3-5
3.2. BOOT SEQUENCE	3-6
3.2.1. DISPLAY	3-6
3.2.2. LOADING THE OPERATING SYSTEM	3-7
3.2.3. FLOW CHART	3-8
3.2.4. ERROR MESSAGES	3-9

4. MENU SYSTEM	4-1
4.1. APPLICATION MENU	4-2
4.2. HOW TO USE THE MENU SYSTEM	4-5
4.2.1. MENU ICONS	4-5
4.2.2. WALKING THROUGH THE MENU SYSTEM	4-5
4.2.3. SELECTING ITEMS	4-6
4.2.4. SPECIAL FUNCTIONS	4-6
4.2.5. SWITCHING FROM ONE CPU TO ANOTHER	4-7
4.2.6. FUNCTION KEYS	4-7
4.3. MAIN UTILITIES MENU	4-8
4.4. EDITOR MENU	4-9
4.5. PRINTING MENU	4-10
4.5.1. PRINT FILES MENU	4-10
4.5.2. PRINTER CONFIGURATION MENU	4-11
4.6. DISK FUNCTIONS	4-15
4.6.1. RENAME FILES MENU	4-15
4.6.2. REMOVE FILE MENU	4-16
4.6.3. COMPARE FILES MENU	4-16
4.6.4. FILESYSTEM PROTECTION MENU	4-17
4.7. COPY FILES MENU	4-18
4.8. DIRECTORY MENU	4-19
4.9. SET TIME MENU	4-21
4.10. HARD/FLOPPY DISK INITIALISATION	4-21
4.10.1. MAKE FILESYSTEM MENU	4-22
4.10.2. FORMAT FLOPPY DISK MENU	4-22
4.10.3. PARTITION HARD DISK MENU	4-23
4.10.4. FORMAT HARD DISK MENU	4-25
4.11. LOADING FROM HARD DISK	4-28
4.11.1. BOOTING FROM THE HARD DISK	4-28
4.11.2. INSTALLING APPLICATION PROGRAMS ON HARD DISK	4-28
5. EDITOR	5-1
5.1. INTRODUCTION	5-1
5.2. STARTING AN EDIT SESSION	5-1
5.2.1. EDIT	5-1
5.2.2. " filename" EDITF	5-1
5.2.3. EDITM	5-1
5.3. OVERVIEW	5-3

5.3.1. EDITOR SCREEN	5-3
5.3.2. CHARACTER - KEYBOARD	5-3
5.3.3. FUNCTION KEYS	5-4
5.3.4. CURSOR KEYS	5-6
5.4. DESCRIPTION	5-7
5.4.1. CHARACTER KEYBOARD	5-7
5.4.2. FUNCTION KEYS	5-7
5.4.2.1. EDITING KEYS	5-7
5.4.2.2. SEARCH- AND REPLACE KEYS	5-9
5.4.2.3. ENTRY MODE KEY	5-10
5.4.2.4. CHARACTER SETS	5-10
5.4.2.5. COLOR EDITING KEYS	5-11
5.4.2.6. BLOCK FUNCTION KEYS	5-12
5.4.2.7. FILE ACCESS	5-13
5.4.2.8. DECOMPRESS FILES	5-13
5.4.2.9. EXIT	5-14
6. ARCHITECTURE	6-1
6.1. HARDWARE	6-2
6.2. MEMORY MAP	6-3
6.3. SOFTWARE	6-4
7. HANDLING OF THE K1197	7-1
7.1. HOUSING	7-1
7.2. MONITOR	7-1
7.3. POWER SUPPLY	7-2
7.4. HARD DISK DRIVE	7-2
7.5. FLOPPY DISK DRIVES	7-2
7.6. BOARDS	7-3
7.6.1. BACKPLANE	7-4
7.6.2. MAIN CPU	7-4
7.6.3. FRONT END CPU (FECp)	7-6
7.6.4. INTERFACE BOARDS AND CONNECTOR MODULES	7-7
7.6.5. CROSSCONNECT BOARD	7-9

8. AUXILIARY INTERFACE	8-1
8.1. REMOTE PORT	8-1
8.1.1. REMOTE SPEED SETTING	8-1
8.1.2. PINOUT	8-2
8.1.3. REMOTE SCREEN DISPLAY	8-3
8.1.4. REMOTE/LOCAL COMMANDS	8-3
8.1.5. REMOTE FUNCTION KEY ENTRY	8-4
8.1.6. REMOTE PLAYBACK OF DATA FILES OR CAPTURE RAM	8-5
8.1.7. TEST MANAGER	8-7
8.2. PRINTER PORT	8-8
8.2.1. PINOUT	8-8
8.2.2. CONNECTING A PRINTER SIEMENS PT88S	8-9
8.3. VIDEO CONNECTOR	8-10
8.3.1. PINOUT	8-10

Program: Operating System for PT K1197
Software Version: V2.2 Rev.1
Previous Version: V2.2 Rev.0
Revision Date: December, 1991

☞ Please read this information carefully before you work with the K1197 and the Operating Instructions. The modifications and extensions described here have a significant influence on the total functions of the K1197.

Installation:

Please exchange the files from disk with the winchester partition WD0.

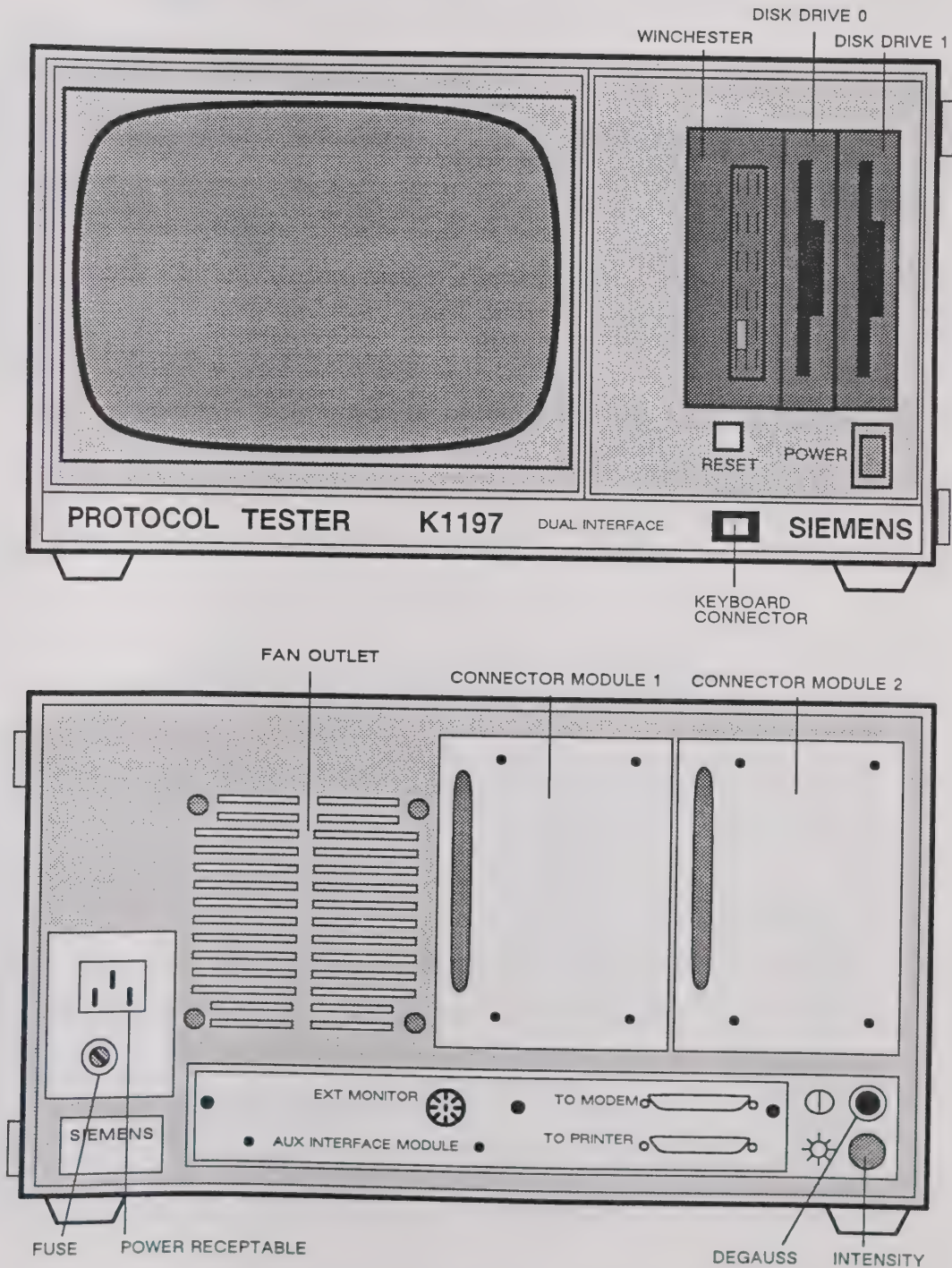
After a reboot the new system is installed.

Changes Made:

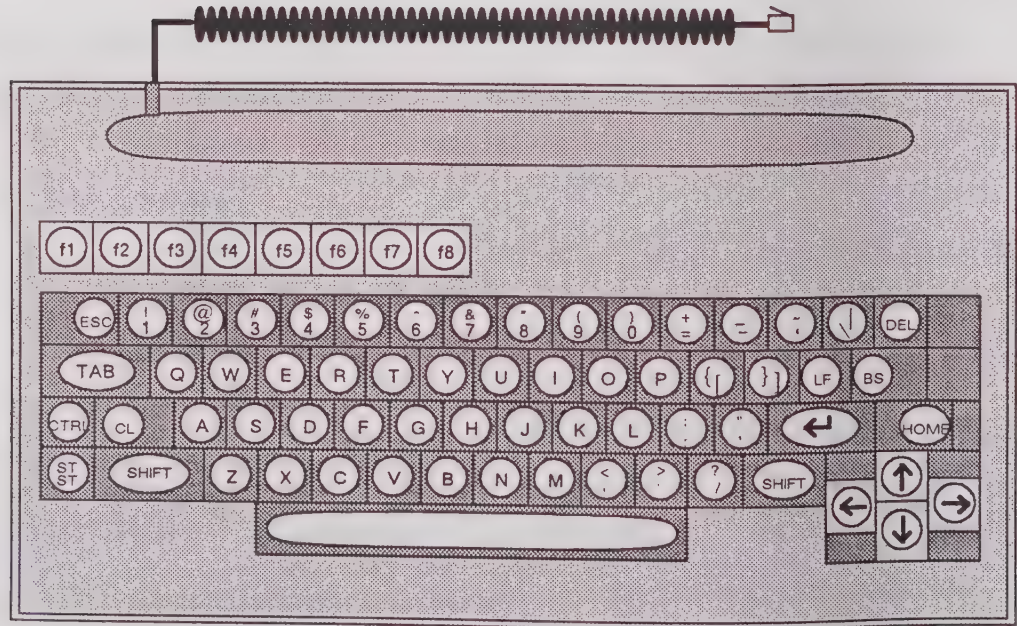
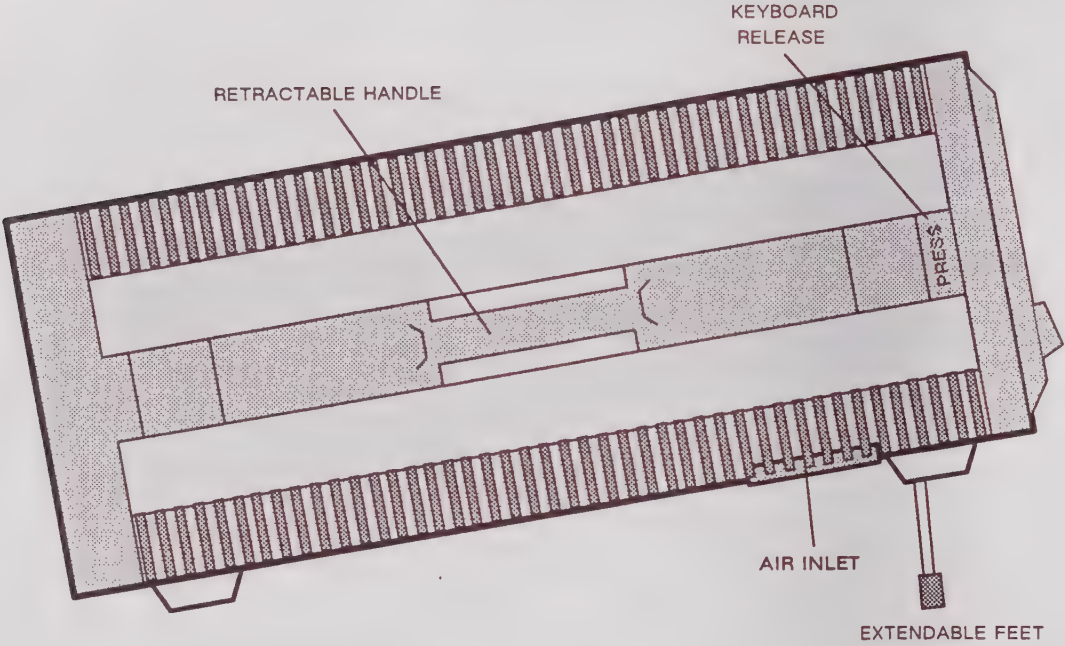
1. The problems upon disk-recording made from any application to a full winchester partion are solved now.
2. There were extremly boot-problems which are solved now.
3. The DEL-key works correctly now.
4. It is not possible to make playback with non-data files anymore.
5. Some new applications are added to the menu selection.

1. GETTING TO KNOW YOUR K1197

1.1. FRONT AND BACK VIEWS



1.2. SIDE VIEW AND KEYBOARD



2. PREPARING FOR OPERATION

2.1. WARNING



WARNING

This equipment should be used according to IEC 536 class 1 safety standard. At operation of the equipment the permanent connection to the ground connector has to be strictly observed.

Before the K1197 is switched on the selected voltage has to be controlled by checking the label near the power receptable.

The fuse must be the type mentioned on the label.

47 - 63 Hz

500 VA max.

FUSE 3,15 A T

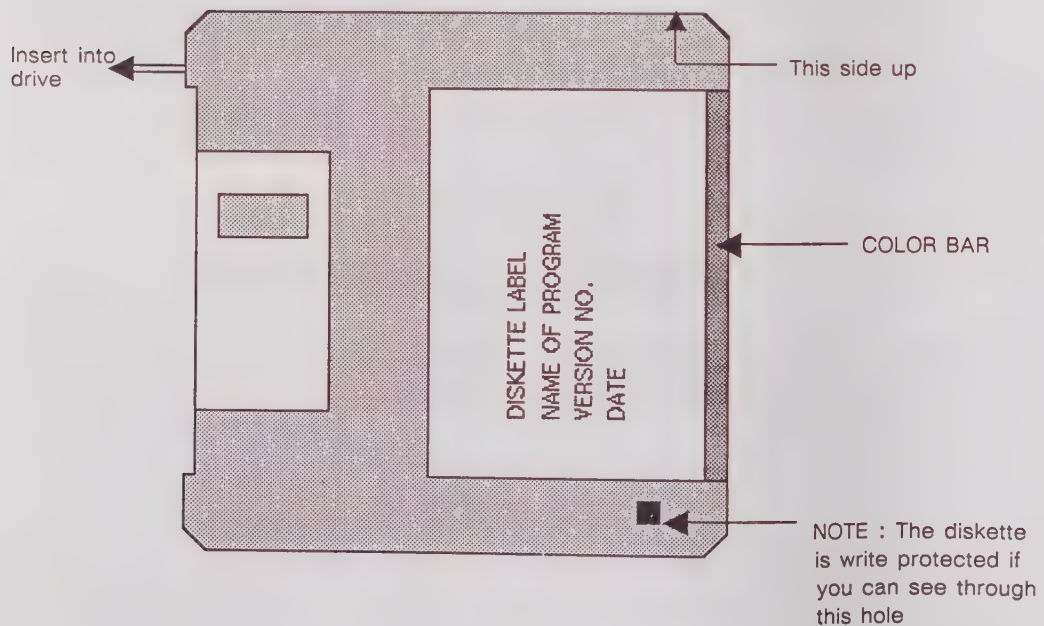


115 V

220 V

2.2. FOR THE FIRST TIME USER

- STEP 1. Use the enclosed power cord, plug the female connector into the power receptable and connect the K1197 to a 110V/220V AC outlet.
- STEP 2. Place unit horizontally and leave a minimum of 1 foot clearance at the back of the K1197.
- STEP 3. Press keyboard release buttons to release keyboard from front bezel. (Plug in keyboard cable to keyboard connector in front bezel if it is not already connected).
- STEP 4. Extendable feet underneath unit at front may be pulled out towards the user for ease in viewing.
- STEP 5. Insert the system diskette into disk drive 0 or 1. The label should face the CRT display. Proceed with chapter 3.



2.3. OPERATING CONDITIONS



CAUTION

Do not obstruct the fan outlet at the back of the K1197.

Do not cover the air inlet at the bottom near the left extendable foot.

Do not operate the K1197 under dusty conditions because no air filters are included.

Do not operate the K1197 under ambient temperatures higher than 40 degrees Celsius (110 degrees Fahrenheit).

3. GETTING INTO OPERATION

Switch on the K1197. The CRT takes approximately 5 seconds to warm up. Then the DIAGNOSTICS and BOOT messages appear. If you see no messages and the screen stays dark, press the reset button **twice for a minimum of two seconds**. Or switch off the K1197 – **wait for about 20 seconds** – and turn the power switch on again.

3.1. POWER ON & RESET DIAGNOSTICS

3.1.1. DISPLAY

When the K1197 is turned on the MAIN CPU is started and it begins running the program of the onboard EPROMs.

This program is only being executed after power up or if the reset button is pressed twice for a minimum of two seconds (cold start).

Pressing the reset button once will start the reloading of the operating system without doing the hardware diagnostics (warm start).

This program needs the following files:

fecp1a.tst, fecp1b.tst, fecp2a.tst, fecp2b.tst and depending on the configuration
bai1.tst, bai2.tst

SIEMENS K1197 Bootstrap 1.0

MAIN CPU DIAGNOSTICS: ROM OK,

INTERFACE 1 APPLICATION PROCESSORS

FECP1 CPU A DIAGNOSTICS: LOADED

FECP1 CPU B LOADING "fecp1b.tst" AT 0xC01000 1540 + 12587008

INTERFACE

INTERFACE 2 APPLICATION PROCESSORS

FECP2 CPU A

FECP2 CPU B

INTERFACE

Harddisk drive is NEC 40Mb

LOADING TEST FILES FROM FLOPPY 0

3.1.2. DIAGNOSTIC TEST

The MAIN CPU checks the configuration of the K1197 and identifies all other CPUs:

- Initialize CRT

- Check ROM Check sum

>>> Display "ROM O.K. <<<

- Identify other CPUs

- Load all other CPUs with test programs

- Start tests on MAIN CPU and all loaded CPUs >

==> All CPUs doing their tests in parallel (see below)

Synchronize all CPUs and do a collision test together

HALT all CPUs ———> STOP

MAIN CPU TEST:

- RAM TEST '1' Uniqueness test

- RAM TEST '2' fill with NOPs

- RAM TEST '3' run the NOPs

- RAM TEST '4' 0xffff pattern

- RAM TEST '5' 0x0000 pattern

>>> Display "1MB RAM O.K." <<<

- TAS instruction test

- HDC write/read test

- DUART 1 write/read test

- DUART 2 write/read test

- HDC interrupt test

- DUART 1 interrupt test

- DUART 2 interrupt test

- CRT interrupt test

- Timer count test

- RTC write/read test

- RTC interrupt test

>>> Display "PERIPHERALS O.K." <<<

FECP TEST:

- RAM TEST '1' 0x12345678 pattern

- RAM TEST '2' uniqueness test

- RAM TEST '3' fill with NOPs

- RAM TEST '4' run the NOPs

- RAM TEST '5' 0xaaaa pattern

- RAM TEST '6' 0x5555 pattern

- RAM TEST '7' 0xffff pattern

- RAM TEST '8' 0x0000 pattern

>>> Display "1MB RAM O.K." <<<

- TAS instruction test

- CIO write/read test

- CIO interrupt test

- USARTs write/read test

- USARTs loopback test with interrupts

- DUART write/read test

- DUART loopback test with interrupts

>>> Display "PERIPHERALS O.K." <<<

BAI2 TEST:

RAM TEST '1' 0x12345678 Pattern
RAM TEST '2' Uniqueness test
RAM TEST '3' fill with NOPs
RAM TEST '4' run the NOPs
RAM TEST '5' 0xaaaa pattern
RAM TEST '6' 0x5555 pattern
RAM TEST '7' 0xffff pattern
RAM TEST '8' 0x0000 pattern

>>> Display "2MB RAM O.K." <<<

TAS instruction test

ICC read/write test

ICC interrupt test

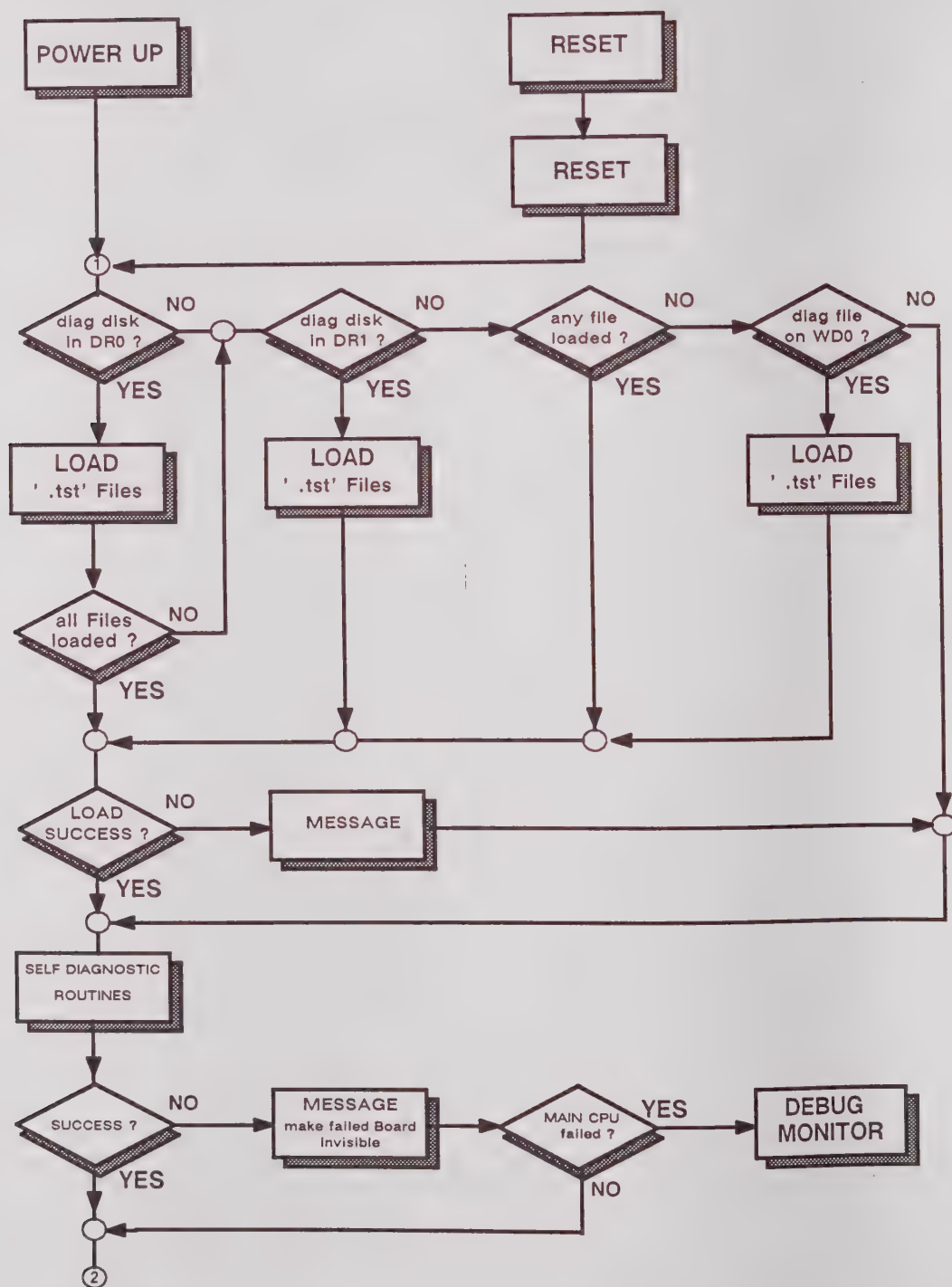
ASIC read/write test

ASIC interrupt test

TIMER interrupt test

>>> Display "PERIPHERALS O.K." <<<

3.1.3. FLOW CHART



3.1.4. FAILURE ON DIAGNOSTICS

If an hardware error is discovered by the diagnostic routines, a simple debug monitor will be invoked. The monitor will allow memory to be displayed and changed and a program to be loaded from floppy drive 0. There are 4 commands available using the CONTROL Keys and the \$ key. In the following discussion, a control key is indicated by the ^ symbol. (i.e. Control B would be represented by ^B)

The "^B" command is used to display bytes starting at a given address. When "^B" is entered the display will print "Display bytes at " and the monitor will wait for an address to be entered.

All numbers are assumed to be in hex. Similarly when "^W" is entered the display will read "Display words at ", and when "^L" is used "Display longwords at " will be displayed. If the entry of one of these commands is in error, a single backspace will delete the entire message, and the monitor will wait for a new command.

Once the command has been entered the following is printed :

(<address>) = ## ,

Where ## is the contents of memory location <address>.

If these contents are to be changed simply enter the new contents in hex terminated by a carriage return. If the contents are not to be changed, then just enter a carriage return. In either case the next memory location will be displayed. If ^L (longword) were selected the next location is 4 bytes past the last location. If ^W (words) were selected the next location is 2 past, and if bytes were selected, the next byte address is used.

The location will continue to advance as long as carriage return is entered. To exit from the command the ESC key is used. In either the ^W or ^L commands, if an odd address is entered, the address will be rounded UP to the next even address. Note that in all display memory commands, leading zeros are not displayed.

To run a program from floppy drive 0, the '\$' command is used, followed by the name of the file to be loaded. The file name given must be identical to that on the disk, and only floppy drive 0 can be used. Once the program is loaded it will be executed. For example the following is used to load the operating system on to the main processor:

\$main.sys

The following is a summary of the debug monitor commands:

- ^B<address> - Display bytes at <address>
- ^W<address> - Display words at <address>
- ^L<address> - Display longwords at <address>
- \$<filename> - will attempt to load and run the file on drive 0 called <filename>.

3.2. BOOT SEQUENCE

3.2.1. DISPLAY

Once the diagnostics are successfully completed, the MAIN CPU starts to load the operating system. Boards which failed in the diagnostics are masked. Software is only loaded to the boards which run thru diagnostics without an error.

This program needs the following files:

fecp1a.sys, fecp1b.sys, fecp2a.sys, fecp2b.sys and depending on the configuration
bai1.sys, bai2.sys

SIEMENS K1197 Bootstrap 1.0

MAIN CPU OPERATING SYSTEM: LOADED

INTERFACE 1 APPLICATION PROCESSORS

FECp1 CPU A OPERATING SYSTEM: LOADED

FECp1 CPU B LOADING "fecp1b.sys" AT 0xC0000 570 + 12582912

INTERFACE

INTERFACE 2 APPLICATION PROCESSORS

FECp2 CPU A

FECp2 CPU B

INTERFACE

Harddisk drive is NEC 40Mb

BOOTING FROM FLOPPY 0

3.2.2. LOADING THE OPERATING SYSTEM

The MAIN CPU performs the following sequence to load the operating system:

1. Check to see if a system disk is in floppy drive DR0 or DR1. If not, proceed to step #3.
2. The disks directories are read and all files ending with ".sys" are loaded. If all files are found, proceed to step #6.
3. Check to see if a system is installed on winchester drive WD0. If not, proceed to step #5.
4. The directory is read and all files ending with ".sys" are loaded. If all files are found, proceed to step #6.
5. The user is prompted to insert a system disk in floppy drive DR0. If a floppy is inserted, proceed to step #1.
6. If "main.sys" is missing, proceed to step #5.
7. The MAIN CPU starts execution at the beginning of "main.sys".

As each file is loaded from the disk a message of the form:

LOADING 'xxx.sys" at OxAAAA TTTTT + DDDDD + BBBBBBBB

"xxx.sys" is the name of the file being loaded

OxAAAA is the address in memory (hexadecimal) where the program is loaded

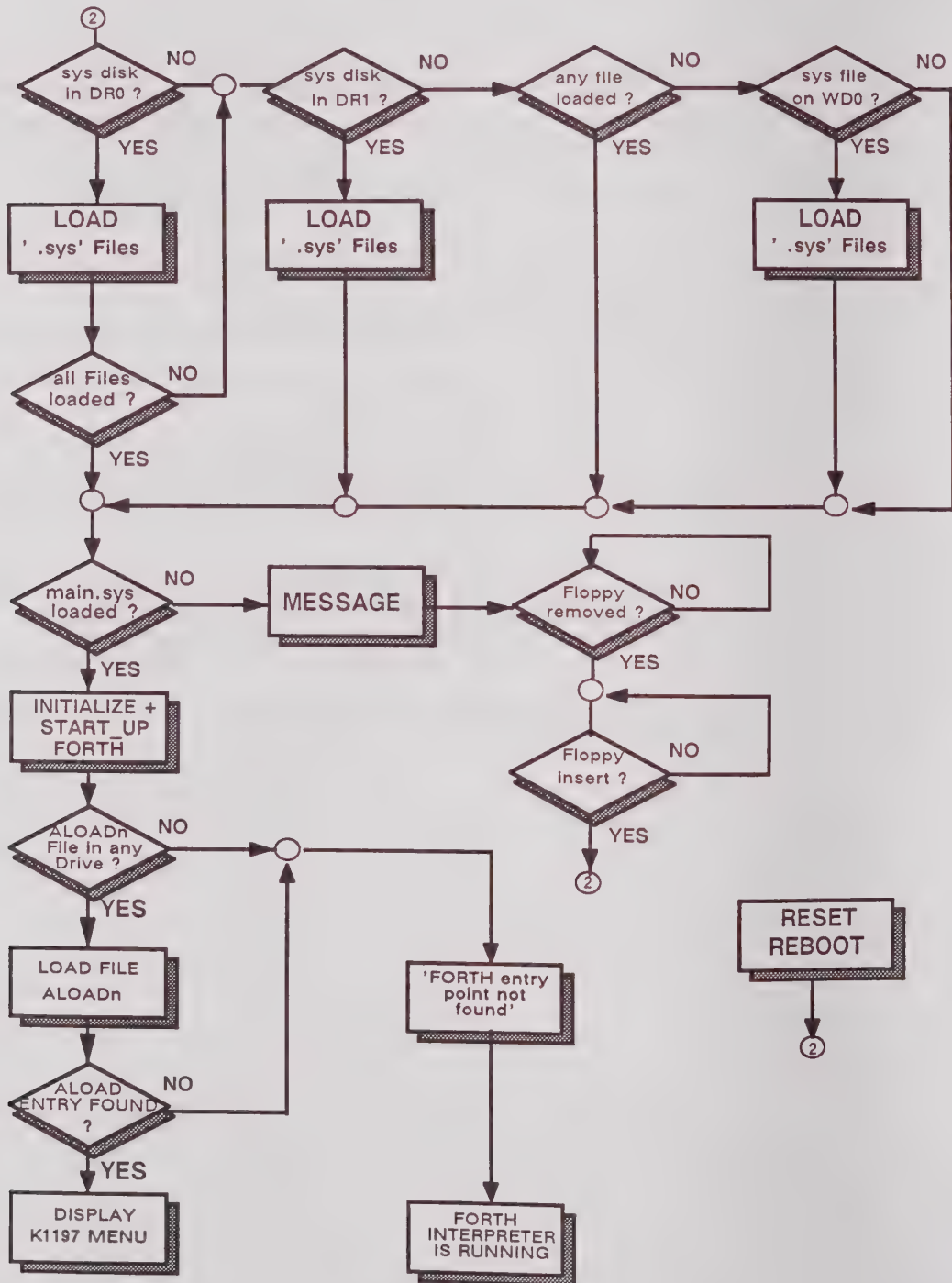
TTTT is the size of the programs text segment (code) in bytes

DDDD is the size of the programs static data segment in bytes

BBBB is the size of the programs BSS (uninitialized data) segment in bytes

A dot '.' is displayed after every disk seek operation to show disk activity (only if booting from floppy disk).

3.2.3. FLOW CHART



3.2.4. ERROR MESSAGES

A disk error may appear during booting. The error is shown in green and red letters. Any disk error detected will cause the last disk operation to be retried until the operation is error free. After 10 unsuccessful retries the entire boot sequence is started from the beginning.

The occurrence of errors during booting using a floppy disk usually indicates work media which should be replaced.

If repeated errors occur when using the hard disk, the user should try booting from floppy and examine his system files on the hard disk. It may be necessary to recopy the system files on WD0 or even to reformat the hard disk. The user should make a backup of all files on the hard disk prior to reformatting.

Format:

```
PPAX: Error P=12CCA a=1 i=2098 | s=0 | e=2081
```

Beschreibung:

PPAX : CPU the error was done.

PPA0 = MAIN CPU

PPA1 = CPU1

PPA2 = CPU2 etc.

```
Error      : Name of error    -- BUS ERROR          ( Bus error, for example: access to
                                   not existing RAM)
                                   -- ZERO DIVIDE         ( divided through zero )
                                   -- ADDRESS ERROR        ( Addressing error, for example:
                                                           Long-access to odd address)
                                   -- ILLEGAL INSTRUCTION   ( not legal instruction )
                                   -- SPURIOUS INTERRUPT     ( unexpected interrupt )
```

P: PROGRAMCOUNTER which contains the instruction that resulted in the error

a: ADDRESS at which was accessed during the aborted Bus cycle

i: Contents of the instruction register

S: STATUS REGISTER

15 14 13 10 9 8 4 3 2 1 0

T	S	I1	I2	I3	X	N	Z	V	C
---	---	----	----	----	---	---	---	---	---

T : TRACE

S : STATUS user/supervisor

I1-I3 : INTERRUPT MASK

e: LOWER ADDRESS

15 5 4 3 2 1 0

		R/W	I/N	function code
--	--	-----	-----	---------------

function code:	001 USER DATA
	010 USER PRGM
	101 SUPERVISOR DATA
	110 SUPERVISOR PRGM
	111 INTERRUPT

Error messages on the Forth-machine have the following look:

The screen turns red but the contents of the screen remains on display. After a RETURN-keypress, an error analysis takes place.

Here is an example caused by tipping on the Forth-Line (window):

```
CPU   DR0   MAIN MENU                                02.08.1989  13:07:33
Address error: PC=0XE093E0   Address=0X1             Instruction code =0X2098
active word: 00E093DA !
called from: 00E0A5AA INTERPRET
Return-stack backtrace:
0X00E0A750 : QUIT  0X00E0CECA : TAKE_ERROR_RETURN  0X00000000 : ?
Stack Trace: Stack was empty
Registers: d0: 00000000  d1: 00000001  d2: 00E36FD9  d3: 00000081  d4:
0000001F
d5: F7FFFFEF  d6: FFFE7DEF  d7: 00000000  a0: 00000001  a1: 00E36FDA  a2:
00E0912E
a3: 00E1C652  a4: 00E0A5AA  a5: 00E093DA  a6: 00E086A8  ssp 00000FE6 usp
00E1Ca36
FIG-FORTH/SIEMENS  MF 1.5  V/X Rev. 3
```

The typed command tries a longword (32 bits) store (!) on an odd address (11).

The 68000 processor allows such manipulation only on even addresses. Thus this error condition is detected as an address error.

The error is caused by the word store (!) which was invoked by the Forth-interpretter. In addition contents of the stack, data and address registers are displayed.

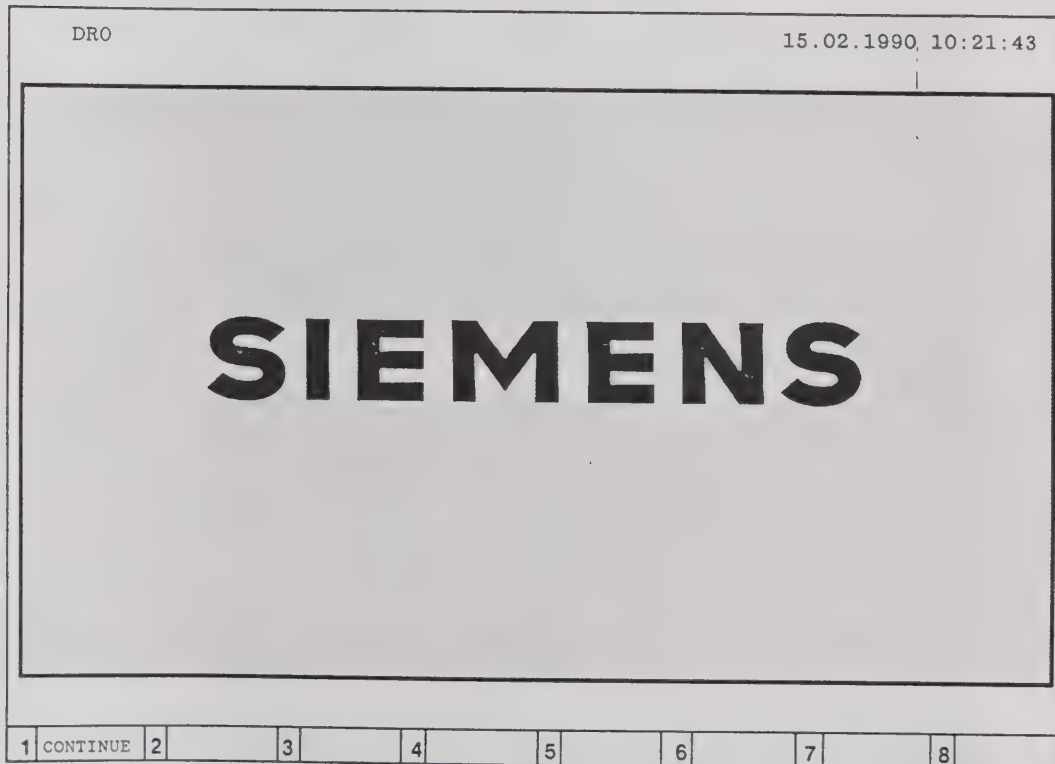
With the contents of the return stack, one can back trace the whole sequence of invocated words. The red screen can be brought back with COME_BACK. The remaindering FORTH-Error messages are described in the FORTH Handbook (-> Message).

4. MENU SYSTEM

This part of the manual describes how the user can select and load application programs and how to reach and use some common utilities, such as file functions and disk drive initialization.

The different selections are described with their corresponding menu windows. Note that all functions can be activated by FORTH commands manually.

After booting the K1197 the following screen will appear:



Please press key f1 to continue.

4.1. APPLICATION MENU

In this menu the installed interface and loaded application program names are displayed. Also new application programs can be loaded any time. **This menu is the first active menu to appear when the K1197 is switched on.**

NOTE: Copying applications from distribution floppy disks to hard disks is explained in chapter 4.11 (Loading From Hard Disk).

The contents of this menu and the possible menu selections depend on the present configuration of interfaces and already loaded programs.

Application programs are divided into two groups: signalling and connection applications. A signalling application is a program that monitors or simulates/emulates a signalling protocol. A connection application is a telecommunication data transfer protocol. An example of the former one is in the ISDN environment the D channel protocol while a B channel protocol would represent the latter one.

DRO		SIEMENS K1197		15.02.1990 10:21:43	
INTERFACE 1:		PRIMARY RATE - S2M			
SIGNALLING APPLICATION:		unloaded ▶			
CONNECTION APPLICATION:		unloaded ▶			
INTERFACE 2:		BASIC ACCESS - S0			
SIGNALLING APPLICATION:		unloaded ▶			
CONNECTION APPLICATION:		unloaded ▶			
unloaded ▶					
OK-M					
OK-M					
1	QUIT_MENU	2	HELP	3	COMMAND
4		5	LOAD CONF	6	SAVE CONF
7		8			

If you want to continue with the general description of the K1197 MAIN MENU, please type cursor left once and proceed with section 4.2. The following example shows how to load an ISDN MONITOR application program working with the second interface installed. Other application programs are loaded accordingly. This example can be followed only, if the second interface installed is a BAI and if a license to use the ISDN D channel monitor has been obtained.

At first, type cursor down two times to highlight the field "unloaded" below the Field "SIGNALLING APPLICATION" in the lower part of the screen. Then type cursor right to get the list of possible application programs. move the cursor up or down until "ISDN MONITOR" is highlighted. Make sure

the disks containing the ISDN MONITOR package are inserted in disk drive DR1 or DR0 (in case this application program has not been copied to the hard disk). Then acknowledge this selection by pressing the return key. The last menu will disappear and the K1197 will start loading the selected application program. Note that the status "unloaded" has been replaced by "loading...". When loading is finished, the message "loading ..." will be replaced by "loaded". The attempt to load a software package may fail, if the software package cannot be found on either floppy disk or hard disk, this is indicated by an error message in the form

Cannot locate "ISDN_TP.B8" or a similar message

Insert disk containing the program to be loaded and try again.

DRO		15.02.1990 10:21:43	
S I E M E N S K 1 1 9 7			
INTERFACE 1:		PRIMARY RATE	
SIGNALLING APPLICATION		SIGNALLING APPLICATION - SO	
CONNECTION APPLICATION		DRIVE: DRO >>	
		ISDN MONITOR	
		ISDN SIMULATION	
INTERFACE 2:		BASIC ACCESS - SO	
SIGNALLING APPLICATION:		unloaded ▶	
CONNECTION APPLICATION:		unloaded ▶	
		unloaded ▶	
OK-M			
OK-M			
1	QUIT_MENU	2	HELP
3	COMMAND	4	
5		6	
7		8	

DRO		15.02.1990 10:21:43	
S I E M E N S K 1 1 9 7			
INTERFACE 1:		PRIMARY RATE - S2M	
SIGNALLING APPLICATION:			
CONNECTION APPLICATION:		SELECT NEW APPLICATION ► ENTER APPLICATION	
INTERFACE 2:		BASIC ACCESS - SO	
SIGNALLING APPLICATION:			
ISDN MONITOR		loaded ►	
CONNECTION APPLICATION:			
		unloaded ►	
		unloaded ►	
OK-M OK-M			
1	QUIT_MENU	2	HELP
3	COMMAND	4	
5		6	
7		8	

If the package has been loaded, type cursor right again, select "ENTER APPLICATION" and acknowledge with the return key. The display will switch to the ISDN MONITOR program.

If the display does not show the standard screen of the newly loaded application, but shows a message like:

This unit is not authorized to run this software package.
Type return to continue.

The reason for a message like this is: an attempt has been made to load a program to a K1197 which has not been licensed for the particular K1197. Type return and then enter the command **MAIN**, followed by return to switch back to the MAIN CPU.

For more details see section 4.11.2.

4.2. HOW TO USE THE MENU SYSTEM

The menu system provides an intuitive interface designed to minimize the need to refer to the manual. It includes pop-up help screens to remind the user of meanings of selections available in the menus.

4.2.1. MENU ICONS

The menus use a number of icons to convey information about the options available on the menu. The borders of the menu are of two types: single line and double line. Double line borders are used to indicate the last menu in a chain. There will not be any menus to the right of a double bordered menu.

At the top and bottom of the right border of a menu there will appear arrows if the menu is bigger than what will fit in the window. The arrow indicates that there are more items outside the window in the direction indicated.

At the right of a selectable item, one of the following may appear: ► or »». The single solid arrow indicates that there is a menu to the right of this item that will be displayed when cursor right is typed. The double arrow indicates that the field takes a number of options which may be scrolled through with the space bar or listed using the cursor right key.

Within a menu, the following keys may be used to move and perform actions:

4.2.2. WALKING THROUGH THE MENU SYSTEM

Cursor up (↑)	Move to next selectable item above this one.
Cursor down (↓)	Move to next selectable item below this one.
Cursor left (←)	Move to previous menu to the left without entering current item. Note: Some menus do not allow this key in order to force the user to select some action through a function key before leaving the menu (e.g: the Hex editor and Message Building System). String and numeric entry fields: Move cursor one character backward.
Cursor right (→)	Move to next menu to the right if one exists. String and numeric entry fields: Move cursor one character forward (first key struck enables editing the already displayed string, which is indicated by a little blinking cursor).
Control cursor up (ctrl-↑)	Move up one page on scrolling menus.
Control cursor down (ctrl-↓)	Move down one page on scrolling menus.
Control shift cursor up (ctrl-shift-↑)	Move to first item in menu.
Control shift cursor down (ctrl-shift-↓)	Move to last item in menu.
Control shift cursor left (ctrl-shift-←)	Close all opened menu windows except the first one.
Control shift cursor right (ctrl-shift-→)	Open all those menu windows which were closed by ctrl-shift-← or a series of cursor left actions.

Home	Move to first item in menu (same as ctrl-shift-↑).
Shift cursor right (shift→)	String and numeric entry fields: Shift the characters from this place on one step forward and create an empty place (blank) between the already existing characters.
Shift cursor left (shift←)	String and numeric entry fields: Delete the character under the cursor and shift the following characters one step backward.
Control cursor right (ctrl→)	String and numeric entry fields: Move the blinking cursor to the last character in the string.
Control cursor left (ctrl←)	String and numeric entry fields: Move the blinking cursor to the first character in the string.
Back Space (BS), Delete (DEL)	String and numeric entry fields: Erase the character before the cursor.
Number and character keys	Search a string from the list of selectable items. Every key struck moves the cursor forward onto the next item to which the string matches. String and numeric entry fields: Erase the old entry and create a new one. If cursor right, ctrl→, shft→ or shft← was pressed first, edit the already existing string.

4.2.3. SELECTING ITEMS

Return	Select an item, complete entry of a typed value or execute a command.
Space bar	When the cursor is on a field which takes a number of options, the space bar may be used to scroll through the available options. These options may also be accessed by typing cursor right and selecting the desired option by pressing Return .

4.2.4. SPECIAL FUNCTIONS



This 3-key combination returns the system to the FORTH interpreter in case of a lock-up or program error.



This 3-key combination prints a hardcopy of the current screen display.

The default printer configuration is:

1200 baud, 8 bits/character, ASCII, DTR and XON/OFF control, LR CR end of line, 80 characters/line, 60 lines/page.

4.2.5. SWITCHING FROM ONE CPU TO ANOTHER

Switching from one CPU to another, that is to change the display and keyboard control to another application program, can be done by pressing the key **ESC**. It toggles the control between application programs (MAIN→CPU1→CPU2→...→MAIN). If there are no applications active on a particular CPU, the next CPU is chosen.

4.2.6. FUNCTION KEYS

There is almost always a set of function keys available on the bottom line of the display. Only the common keys for all application programs are listed here.

QUIT_MENU / MENU	: Leaves the menu system control / back to menu system control
HELP / QUIT_HELP	: Open a help menu for the particular menu window.
COMMAND /	: Give the control to the FORTH interpreter which enables the use
QUIT_COM	of keyboard for typing commands. The menu system window remains still visible. Return the control back to the menu system.

4.3. MAIN UTILITIES MENU

DRO		SIEMENS K1197		15.02.1990 10:21:43	
◆◆ UTILITIES ◆◆					
Application selection ▶ Editor ▶ Printing ▶ Disk functions ▶ Copy files ▶ Directory ▶ Set time ▶ Hard/floppy disk init. ▶					
OK-M					
OK-M					
1	QUIT MENU	2	HELP	3	COMMAND
4		5	LOAD CONF	6	SAVE CONF
7		8			

MENU SELECTION:

- Application selection** : Select the K1197 APPLICATION MENU in which all application program selections made and interfaces installed are shown. Loading application program is made in this menu.
- Editor** : Select EDITOR MENU for editing files.
- Printing** : Select PRINTER CONFIGURATION and PRINT FILES MENU.
- Disk functions** : Open a new menu to select menus to rename, remove and compare files and to (re)set file system write protection.
- Copy files** : Select COPY FILES MENU which provides the functions to copy files.
- Directory** : Select DIRECTORY MENU to display floppy or hard disk contents.
- Set time** : Open SET TIME MENU to adjust the real time clock.
- Hard/floppy disk init.** : Open a menu to select floppy and hard disk initialisation functions (ie. formatting floppy/hard disk, make filesystems).

FUNCTION KEYS:

- LOAD CONF** : Open a menu for loading a previously saved application program configuration and restoring a printer configuration.
- SAVE CONF** : Open a menu for saving currently loaded application programs so that they can be loaded again later on with LOAD CONF. Saves also the current printer configuration.

4.4. EDITOR MENU

DRO	SIEMENS K1197	15.02.1990 10:21:43
◆◆ UTILITIES ◆◆		
Application s Editor Printing Disk function Copy files Directory Set time Hard/floppy disk init.	DRIVE: DRO: >> FILE NAME: EDIT FILE EDIT BUFFER	
OK-M OK-M		
1	2	3
QUIT_MENU	HELP	COMMAND
4	5	6
7	8	9

MENU SELECTION:

DRIVE: : Select disk drive.

FILE NAME: : Enter file name if so desired.

EDIT FILE : Read above entered file to EDITBUF for editing the file. If the field for the file name is empty, EDIT program prompts for a file name.

EDIT BUFFER : Read the contents of EDITBUF for editing. No file operations are performed. This is command useful when EDIT was left without saving its contents.

Note: Some other disk operations (eg. file printing) use also EDITBUF, which changes its contents totally.

FUNCTION KEYS:

HELP : Open a help menu.

COMMAND : Open FORTH interpreter window.

4.5. PRINTING MENU

4.5.1. PRINT FILES MENU

DRO	SIEMENS K1197	15.02.1990 10:21:43					
◆◆ UTILITIES ◆◆							
Application sel Editor Printing Disk functions Copy files Directory Set time Hard/floppy disk init.	Print files Printer Conf	DRIVE: DRO: >> FILE NAME: PRINT FILE(S)					
OK-M OK-M							
1	2	3	4	5	6	7	8
QUIT_MENU	HELP	COMMAND					

MENU SELECTION:

DRIVE: : Select disk drive.
FILE NAME: : Enter file name. Allowed wild cards: * ? []
PRINT FILE(S) : Execute file printing. The files to be printed are also displayed when printing.

FUNCTION KEYS:

HELP : Open a help menu.
COMMAND : Open FORTH interpreter window.

4.5.2. PRINTER CONFIGURATION MENU

DRO	SIEMENS K1197	15.02.1990 10:21:43
◆◆ UTILITIES ◆◆		
Application se Editor Printing Disk functions Copy files Directory Set time Hard/floppy disk init. Print files Printer Config	PRINTER CONFIGURATION Printer speed: 1200 >> Control mode: DTR&XOF >> Display format: CHAR >> End of line: LFCR >> Chars/line: 80 Lines/page: 60	
OK-M		
OK-M		
1 QUIT_MENU	2 HELP	3 COMMAND
4	5	6
7	8	

MENU SELECTION:**Printer speed:**

Set printer speed. Scroll through the existing alternatives by pressing the space bar until the correct value is found or by opening the complete selection menu (cursor right) and making the selection there (Return).

Control mode:

: Set control mode as described above.

Display format:

: Set display format as described above.

End of line:

: Set end of line character as described above.

Chars/line:

: Enter number of chars/line by typing the wanted value.

Lines/page:

: Enter number of lines/page by typing the wanted value.

FUNCTION KEYS:**HELP**

: Open a help menu.

COMMAND

: Open FORTH interpreter window.

The printer may be configured either from the PRINTER CONFIGURATION MENU or by using the following FORTH words.

PRINTER BAUD RATE

P_BAUD=50	(--) Sets printer baud rate to 50
P_BAUD=75	(--) Sets printer baud rate to 75
P_BAUD=110	(--) Sets printer baud rate to 110
P_BAUD=134	(--) Sets printer baud rate to 134
P_BAUD=150	(--) Sets printer baud rate to 150
P_BAUD=200	(--) Sets printer baud rate to 200
P_BAUD=300	(--) Sets printer baud rate to 300
P_BAUD=1200	(--) Sets printer baud rate to 1200
P_BAUD=1800	(--) Sets printer baud rate to 1800
P_BAUD=2000	(--) Sets printer baud rate to 2000
P_BAUD=2400	(--) Sets printer baud rate to 2400
P_BAUD=9600	(--) Sets printer baud rate to 9600

PRINTER CONTROL

CONTROL=OFF	(--) Printer control turned off
CONTROL=DTR	(--) DATA TERMINAL READY control
CONTROL=XOF	(--) XON/XOFF control
CONTROL=DTR&XOF	(--) Both of above

PRINTER DISPLAY FORMAT

CHAR_PRINT_MODE	(--)	Standard output mode, checks for non-printable characters and outputs these in format <AB>, where A,B are the two nibbles represented in HEX notation.
RAW_PRINT_MODE	(--)	Outputs all characters without checking for non-printable characters.
HEX_PRINT_MODE	(--)	Outputs all characters in HEX format <AB>, where A,B are the two nibbles represented in HEX notation. The string "HALLO" will be printed as: <48><41><4C><4C><4F>.

PRINTER END OF LINE

EOL=NONE	(--)	Carriage return not generated automatically.
EOL=CR	(--)	Carriage return is generated.
EOL=LFCR	(--)	Line feed is generated, followed by a carriage return.

CHARACTERS/LINE

Change of characters/line must be done in the PRINTER CONFIGURATION MENU.
If characters/line is equal to null no range check is performed.
For printing source files made K1197 editor the value should be 80 chars/line.

LINES/PAGE

Change of lines/page must be done in the PRINTER CONFIGURATION MENU).

Printer output can be created and sent to the printer by using the following group of FORTH words. The words that are used for creating and printing are: PEMIT, P.TYPE, P." ", P., P.H, PCR, and PRT.

PEMIT	(value --) PEMIT will insert the character value passed to it on the FORTH stack into the current position of the printer buffer. The PEMIT operates for the printer output like EMIT does for command lines.
P.TYPE	(addr / count --) P.TYPE will insert the character string starting at address with a length of count into the current position of the printer output buffer. The P.TYPE operates for printer output like TYPE does for command lines.
P."	(--) P." inserts a character string into the printer buffer. P." is used in the form: P." text string", where, the terminating delimiter is a " (quote). The P." operates for trace reports like ." does for command lines.
P.	(value --) P. performs a binary to ASCII conversion on the value that is on the FORTH stack and inserts the result followed by one space into the printer buffer. BASE is used to determine the number base for the conversion with decimal being the default. The P. operates for printer output like . does for command lines.
P.H	(value --) P.H performs a binary to ASCII conversion using HEX base on the value that is on the FORTH stack and inserts the result followed by one space into the printer buffer. The P.H operates for printer output like .H does for command lines.
PCR	(--) PCR outputs the current printer buffer followed by the selected end of line characters. The PCR operates for trace reports like the CR does for command lines.
PRT	(--) PRT outputs the current printer buffer. No end of line characters are sent.
PRINT_FF	(--) PRINT_FF outputs a form feed.

4.6. DISK FUNCTIONS

DRO	S I E M E N S K 1 1 9 7	15.02.1990 10:21:43
◆◆ UTILITIES ◆◆		
Application selection ▶ Editor Printing Disk function ▶ Copy files Directory Set time Hard/floppy disk init. ▶ Rename file ▶ Remove file ▶ Compare files ▶ File system protection ▶		
OK-M		
OK-M		
1	2	3
QUIT MENU	HELP	COMMAND
4	5	6
7	8	

4.6.1. RENAME FILES MENU

MENU SELECTION:	
DRIVE:	: Select disk drive.
OLD NAME:	: Enter source file name.
NEW NAME:	: Enter destination file name.
RENAME FILE:	: Execute renaming.
FUNCTION KEYS:	
HELP	: Open a help menu.
COMMAND	: Open FORTH interpreter window.

4.6.2. REMOVE FILE MENU

MENU SELECTION:

DRIVE: : Select disk drive.
FILE NAME: : Enter file name. Allowed wild cards: * ? []
REMOVE FILE: : Display all file names that are to be removed and when positively acknowledged, also remove them.

FUNCTION KEYS:

HELP : Open a help menu.
COMMAND : Open FORTH interpreter window.

4.6.3. COMPARE FILES MENU

MENU SELECTION:

DRIVE: : Select disk drive.
COMPARE FROM: : Enter first file name. Allowed wild cards: * ? []
COMPARE TO: : Enter second file name.
COMPARE FILES : Execute file compare. Shows if the files are identical or not.

FUNCTION KEYS:

HELP : Open a help menu.
COMMAND : Open FORTH interpreter window.

4.6.4. FILESYSTEM PROTECTION MENU

It is possible to set a write protection to a whole filesystem, that is, to a disk drive or a logical hard disk partition. When the protection is on, the files in the protected filesystem can be only read but no changes in the contents can not be stored.

MENU SELECTION:	
DRIVE:	: Select disk drive.
PROTECT FILESYSTEM	: Set write protection to the filesystem.
UNPROTECT FILESYSTEM	: Reset write protection from the filesystem.
FUNCTION KEYS:	
HELP	: Open a help menu.
COMMAND	: Open FORTH interpreter window.

4.7. COPY FILES MENU

DRO	S I E M E N S K 1 1 9 7	15.02.1990 10:21:43													
◆◆ U T I L I T I E S ◆◆															
Application sele Editor Printing Disk functions Copy files Directory Set time Hard/floppy disl	COPY FROM: DRIVE: DRO: >> FILENAME: COPY TO: DRIVE: DRO: >> FILENAME: WITH COMPARE: OFF >> COPY FILE(S)														
OK-M															
OK-M															
1	QUIT_MENU	2	HELP	3	COMMAND	4		5		6		7		8	

MENU SELECTION:

DRIVE: : Select disk drive.
 COPY FROM: : Enter source file name. Allowed wild cards: * ? []
 COPY TO: : Enter destination file name.
 WITH COMPARE: : When selected, COMPARE is executed after copying.
 COPY FILE(S) : Execute file copying.

FUNCT ION KEYS:

HELP : Open a help menu.
 COMMAND : Open FORTH interpreter window.

4.8. DIRECTORY MENU

DRO	SIEMENS K1197	15.02.1990 10:21:43													
◆◆ UTILITIES ◆◆															
Application selection Editor Printing Disk functions Copy files Directory Set time Hard/floppy disk init.	DRIVE: DRO: >> FILENAME: * DIRECTORY SHORT DIRECTORY LONG														
OK-M OK-M															
1	QUIT_MENU	2	HELP	3	COMMAND	4		5		6		7		8	

MENU SELECTION:

- DRIVE:** : Select disk drive.
- FILE NAME:** : Enter file name. Allowed wild cards: * ? []
- DIRECTORY SHORT** : Display the contents of the selected floppy/hard disk drive in short format (file name).
- DIRECTORY LONG** : Display the contents of the selected floppy/hard disk drive in long format (file name, type, size, status and date of last change).

Note: If the list of files is longer than the window size, the directory contents can be scrolled with the cursor keys. Directory window is left by pressing cursor left.

FUNCTION KEYS:

- HELP** : Open a help menu.
- COMMAND** : Open FORTH interpreter window.

Filename Entry

Filenames may be preceded with the drive designation DR0: and DR1:, WD0:, WD1: to WD7: if the current partition supports that logical drive.

eg. DR0:SOURCE1
DR1:SOURCE2

Wildcard entry permits the use of the following meta characters: *, ?, [and].

* – any character sequence or no characters

? – any single character

[] – delimiters to allow a choice of characters or a range of

characters. e.g.

[123] choice of 1, 2 or 3.

[1-3] range of 1 to 3.

[a-zA-Z] range of a to z or A to Z

Filenames may contain wildcards for the following operations:

DIRECTORY SHORT
DIIRECTORY LONG
REMOVE
PRINT
COPY

Care must be taken in using the COPY operation. The following examples show typical usage.

SOURCE	DESTINATION
DR0:*	DR1:
DR0:SOURCE1	DR0:SOURCE2
DR0:SOURCE1	DR1:SOURCE1
DR0:SOURCE[123]	DR1:
DR1:SOURCE?.F	DR0:

It should be obvious from the example chosen, that a destination filename may be entered for a single file only. If multiple files are copied, the destination files will have the same names as the source files.

4.9. SET TIME MENU

The real time clock in the K1197 is battery backed up. In case there is a need to adjust the time despite of the battery protection, the time can be set in the SET TIME MENU. The time is typed as numbers in the corresponding fields and then the cursor is moved to SET TIME and Return pressed. When the time is set, the control is given back to the MAIN UTILITIES MENU.

4.10. HARD/FLOPPY DISK INITIALISATION

DRO

SIEMENS K1197

15.02.1990 10:21:43

◆◆ UTILITIES ◆◆

Application selection ▶

Editor ▶

Printing ▶

Disk function

Copy files

Directory

Set time

Hard/floppy

Make filesystem ▶

Format diskette ▶

Partition hard disk ▶

Format hard disk ▶

OK-M

OK-M

1

QUIT_MENU

2

HELP

3

COMMAND

4

5

6

7

8

4.10.1. MAKE FILESYSTEM MENU

MENU SELECTION:

DRIVE: : Select disk drive.
VOLUME NAME: : Enter name for the filesystem.
MAKE FILESYSTEM : A warning is given that all files will be lost on the selected disk drive and an acknowledgement is required before the execution of making a filesystem.

FUNCTION KEYS:

HELP : Open a help menu.
COMMAND : Open FORTH interpreter window.

4.10.2. FORMAT FLOPPY DISK MENU

MENU SELECTION:

DRIVE: : Select disk drive.
DISK TYPE: : Choose disk type: High Density or Double Density.
FORMAT FLOPPY DISK: A warning is given that all files will be lost on the selected disk drive and an acknowledgement is required before the disk is formatted. A filesystem is automatically created with the volume name 'K1197_DISK'.

FUNCTION KEYS:

HELP : Open a help menu.
COMMAND : Open FORTH interpreter window.

Formatting Diskettes

The 'FORMAT FLOPPY DISK' operation formats the diskette for the selected floppy drive and automatically makes a file system on the diskette with the volume name K1197_DISK. The user may perform the 'MAKE FILESYSTEM' separately, for example in order to change the volume name.

4.10.3. PARTITION HARD DISK MENU

Hard disk partitioning divides the Winchester into smaller logical hard disk drives (WD0, WD1, WD2...).

On top of the PARTITION HARD DISK MENU there is displayed the size of the whole Winchester. The user can give the desired logical drive sizes by pressing cursor right when the cursor points the text 'WDx'. A new menu is opened, and the user can type the wanted size of the chosen hard disk drive in blocks (a block = 512 bytes) or just press Return on the 'REMAINING blocks' which gives this partition all remaining blocks. If there is still space left on the hard disk after the made selection, a new WDx+1 is created. In this way it is possible to repeat the same procedure for up to eight logical disk drives.

EXECUTE PARTITION can be pointed with the cursor any time and executed with Return. If there is still space left on the hard disk, the WDx with the biggest sequence number x gets the rest of the blocks.

If there are two Winchester drives installed in the K1197, partitioning is executed to the selected one, that is, when the active disk drive is WD0..WD7, hard disk drive 0 is the target Winchester, and when WD10..WD17 is selected, the partitioning is executed in the Winchester drive 1. However, if either floppy disk drive was chosen, WD0 is changed to be the active drive regardless of the amount of existing Winchesters.

DRO	SIEMENS K1197	15.02.1990 10:21:43													
◆◆ UTILITIES ◆◆															
Application selection Editor Printing Disk functions Copy files Directory Set time Hard/floppy disk Make Forma Parti Forma	Total size of Winchester drive 0: 41088 kbytes (82176 blocks) ENTER SIZE OF PARTITION (BLOCKS): WD0: 10000 ► WD1: 5000 ► WD2: 5000 ► WD3: ► EXECUTE PARTITION														
OK-M OK-M															
1	QUIT_MENU	2	HELP	3	COMMAND	4		5		6		7		8	

MENU SELECTION:**WDx**

: Select a menu for entering the size of WDx. When the size is defined and there is still space left on the hard disk, a new place (WDx+1) is created. Already made selections can be changed without any limitations before EXECUTE PARTITION is executed.

EXECUTE PARTITION

: A warning is given that all files will be lost on the selected hard disk and thus, an acknowledgement is required before the partitioning is completed. Filesystems will be created automatically.

FUNCTION KEYS:**HELP**

: Open a help menu.

COMMAND

: Open FORTH interpreter window.

4.10.4. FORMAT HARD DISK MENU

This menu works and looks just like PARTITION HARD DISK MENU except the text EXECUTE PARTITION is replaced by FORMAT HARD DISK and before partitioning the hard disk will be formatted.

WINCHESTER HARD DISK SUPPORT

The K1197 has a 40 MByte Winchester disk drive installed. The K1197 software will automatically detect the number of floppy and Winchester drives presently installed, and configure itself accordingly.

Organization of a Hard Disk

A physical hard disk drive can be divided into a number of logical partitions or 'logical disks'. Therefore, a logical disk could either be an entire physical disk or only part of an entire disk.

It is possible for the physical disk to be partitioned into (between 1 to 8) logical disks.

The names used for the logical disks are:

WD0, WD1, WD2, WD3, WD4, WD5, WD6, WD7

These names only apply if the physical disk is currently partitioned to support the corresponding logical disk. That is, if the physical disk is divided into 4 parts, only WD0 through WD3 would be valid.

Consequently, the names WD4 through WD7 would generate a "Non-existent I/O device" error message.

System Regeneration



CAUTION

The following procedure destroys all software files present on the hard disk.
This procedure should only be necessary in the event of a hard disk failure, or total accidental erasure of the hard disk.

This procedure is used to:

1. Format and re-partition the hard disk.
2. Restore Operating System files.
3. Restore Application Software.
4. Restore customer source file(s).

When shipped, the hard disk in the K1197 is divided into four parts, i.e. WD0, WD1, WD2 and WD3.

WD0 is reserved for the system files.

WD1 is normally used for application software.

WD2 is normally used for customer source files.

WD3 is normally used for data recording.

This is the recommended number of partitions for the hard disk.

When used as a protocol development tool, the user may wish to change the number of partitions.

STEP 1 Format the Hard Disk

Insert the Operating System Disk(s) into Floppy Drive(s) 0 (and 1).

Power on the unit and press "RESET".



CAUTION

The following command will completely destroy the contents of the hard disk.

Now use the FORMAT HARD DISK MENU or type the commands:

```
WD0 NO WRTPTRT_FS
```

```
WD1 NO WRTPTRT_FS
```

```
WD2 NO WRTPTRT_FS
```

```
WD3 NO WRTPTRT_FS
```

```
2000 4000 5000 3 FORMATWD
```

This will create four logical disks with the sizes shown above. The fourth (last) logical disk has the remaining size of the physical disk and type:

```
WD0 " WD0" MAKE_FS
```

```
WD1 " WD1" MAKE_FS
```

```
WD2 " WD2" MAKE_FS
```

```
WD3 " WD3" MAKE_FS
```

The sizes shown above are the number of 512 byte blocks for each logical disk. The total number of blocks available on a Winchester disk is approximately 40000 blocks.

STEP 2 Restore Operating System Files

```
DRO " *" " WD0;" 2DUP COPY COMPARE
```

Confirm that no errors have occurred during the copy operation.

STEP 3 Restore Application Software

Insert the disks containing Application Software backups in Floppy Drive DR0 and type:

```
DR0 " *" " WD1:" 2DUP COPY COMPARE
```

Confirm that no errors have occurred during the copy operation.

Repeat this step until all the software is restored.

STEP 4 Restore Customer Source Files

Insert the disk containing local source files into Floppy Drive DR0 and type:

```
DR0 " *" " WD2:" 2DUP COPY COMPARE
```

Confirm that no errors have occurred during the copy operation.

Repeat this step until all the sources are restored.

STEP 5 Write Protect the Filesystem (if required)

```
WD0 YES WRTprt_FS  
WD1 YES WRTprt_FS  
WD2 YES WRTprt_FS
```

R E S T O R E C O M P L E T E !

Remove all floppy disks and reboot the K1197.

4.11. LOADING FROM HARD DISK

4.11.1. BOOTING FROM THE HARD DISK

If your K1197 does not boot from hard disk or if you want to replace the existing copy of the operating system with a new one, insert the new K1197 system disks into drive DR0 and DR1. Then select the COPY FILES MENU and copy the entire contents of disks DR0 and DR1 to the hard disk partition WD0 (source disk = DR0/DR1, source file name = * , destination disk = WD0, destination filename = <empty>). Note that only WD0 is a bootable hard disk partition. If a copy of the operating system is saved on other partitions than WD0, they will not be considered bootable.

4.11.2. INSTALLING APPLICATION PROGRAMS ON HARD DISK

By default application packages are shipped on floppy disks. The programs on these disks can be loaded directly from floppy disks. However, this takes considerable time. It is recommended to copy the contents of these floppy disks to the hard disk, for instance to partition WD1 (leaving WD0 for the operating system). Use the copy files menu to copy the files from floppy disk to hard disk.

With every application package comes a file named **KEY_FILE**. This file contains codes indicating a license for a specific K1197 application package has been issued. A software package can only be loaded to the K1197, if a an appropriate code is found in **KEY_FILE**. When loading application programs, the file **KEY_FILE** is searched for first on the disk or partition, where the software is loaded from, then on hard disk partition WD0. If more than one application program is copied to the K1197's hard disk, the several **KEY_FILES** have to be merged into one, which should be stored on partition WD0. Merging files is done with the inbuilt full screen editor. Select the Edit File menu, select partition WD0, filename **KEY_FILE** and start the editor (EDIT FILE). Use the INS FILE function (insert file) to merge **KEY_FILES** from different distribution floppy disks. Then reselect WD0 and save the file with the name **KEY_FILE**.

5. EDITOR

5.1. INTRODUCTION

The K1197-Editor is a screen editor which may be used to create or change files. The Editor is implemented in a way, that all features are accessible by use of function keys.

The Editor works on a static buffer of 64 kBytes. Note that the Editor in this software release uses a so called "compressed" file format which is incompatible to the file format of previous software releases. The Editor will automatically convert old files formats into the new one. However you want to convert this compressed file format in a old format (i.e. suitable to older systems), the FORTH - Command **DECOMPRESS_FILE** (see 5.4.2.8.) must be used.

In the following documentation the functions and their handling are explained.

5.2. STARTING AN EDIT SESSION

Three FORTH-Commands will start an edit session.

5.2.1. EDIT

In this mode the Editor is started interactively. It is intended for general purpose editing as in program development.

The FORTH word **EDIT** supports this mode.

After entering the Editor the user is asked for the filename.

The drive should be selected by pressing the DR0, DR1, WD0, .. , WD7 toggling function key. This can also be done by entering the filename, preceeded by "DR0:..." , "DR1:..." , "WD0:..." ... respectively.

The selected file will be loaded and some internal operations will take place to guarantee, that the file has the correct format. If the entered filename does not exist for the selected drive the message "`<filename> [New File]`" will be displayed.

5.2.2. " filename" EDITF

The Editor can also be started entering the filename as a FORTH-String Parameter. Followed by the FORTH command **EDITF**. The specified file will be automatically loaded.

Example : "**TEST.F**" **EDITF**

5.2.3. EDITM

Leaving the editor accidentally will cause the loss of all edited data. To recover from this situation the **EDITM** command may be used. The command **EDITM** will reconstruct the status of the Editor before it was left the last time.

Note that all data is completely lost after performing any disk operation or calling **EDITOR** using **EDIT** or **EDITF** once more.

Edit:MY_FIRST_FILE
Row: 1
Column: 1
CS: ASCII
CHAR: ,1988-01-01 12:00:00

This is the first line in the file

End of File

1	MARK	2	UNMARK	3	BLK MOVE	4	BLK COPY	5	BLK DEL	6	BLK SAVE	7	INS FILE	8	CS:ASCII
1	<next>	2	DRO	3	JOIN LIN	4	NEW FILE	5	REVERSE	6	SET ATTR	7	MOD:MISC	8	NORMAL
1	INS/REP	2	INS LINE	3	CUT CHAR	4	CUT LINE	5	PASTE	6	SAVE	7	SRCH/REP	8	QUIT

Figure 5-1: Editor Screen

Explanation to figure 5-1 :

Line 1	- Status Line				
Line 2 - 19	- Text Display				
Line 20	- Command Line(User input, Error messages)				
Line 21	- Function Keys	CTRL	f1	- f8	
Line 22	- Function Keys	SHIFT	f1	- f8	
Line 23	- Function Keys		f1	- f8	

5.3. OVERVIEW

5.3.1. EDITOR SCREEN

- Status Line (Line 1)

The first line on the screen is the status line. It shows the name of the file that the EDITOR is working on, the actual cursor position in terms of row and column number, the character set (ASCII e.g.) of the character at the cursor, and the HEX value of that character. A inverted bar display in the status line shows the filling percentage of the 64 kByte EDITOR buffer. A fully inverted bar on the status line would have the meaning, that 100% of the Editor buffer is used.

- Text Display (Line 2-19)

The lines 2 - 19 display a part of the edited file where the user is able to input text and make changes.

- Command Line (Line 20)

Under special circumstances the user is prompted for input (e.g. saving a file needs the input of the file name). This line is also be used by the EDITOR to output ERROR MESSAGES and WARNINGS.

- Function Keys (Line 21-23)

Show the actual definitions of the function keys.

5.3.2. CHARACTER - KEYBOARD

The character keyboard, including the tab, return, and backspace keys, allows a user to completely manipulate text with capabilities like a typewriter. If, while entering text, the end of a line is reached, no more characters are allowed and an error message is displayed. The tab key inserts up to four spaces to the next tab position which is 5, 9, 13, e.g.

The action of all of these keys depends on the insert/replace key (Bottom Bank key f1) . When in insert mode, the return key causes the insertion of a new line, and tab causes subsequent text on a line to be moved over. When in replace mode, the return key replaces the current character with a new line.

Note that when the cursor is in the middle of a line of text, a backspace moves subsequent text over one space to the left and removes the character previously in that space in insert mode. However a backspace replaces the character by a space and moves the cursor to the left in replace mode.

5.3.3. FUNCTION KEYS

The function keys provide the majority of the extra functions expected in an screen oriented editor. There are three banks of function keys, most of them are self explaining.

1 Mark	2 Unmark	3 Blk Move	4 Blk Copy	5 Blk Del	6 Blk Save	7 Ins File	8 CS:ASCII
1 (next)	2 DRO	3 JoinLine	4 New File	5 Reverse	6 Set Attr	7 Mod:Misc	8 Normal
1 INS/REP	2 INS Line	3 Cutchar	4 Cut Line	5 Paste	6 Save	7 Srch/Rep	8 Quit

- Editor keys

1 Mark	2 Unmark	3 Blk Move	4 Blk Copy	5 Blk Del	6 Blk Save	7 Ins File	8 CS:ASCII
1 (next)	2 DRO	3 JoinLine	4 New File	5 Reverse	6 Set Attr	7 Mod:Misc	8 Normal
1 INS/REP	2 INS Line	3 Cutchar	4 Cut Line	5 Paste	6 Save	7 Srch/Rep	8 Quit

INS/REP toggles insert/replace mode.

Join line joins two lines, as long as they fit in one line.

Cut and Paste allows easy deleting and moving.

- Insert/Replace keys

1 Mark	2 Unmark	3 Blk Move	4 Blk Copy	5 Blk Del	6 Blk Save	7 Ins File	8 CS:ASCII
1 (next)	2 DRO	3 JoinLine	4 New File	5 Reverse	6 Set Attr	7 Mod:Misc	8 Normal
1 INS/REP	2 INS Line	3 Cutchar	4 Cut Line	5 Paste	6 Save	7 Srch/Rep	8 Quit

Strings must be searched and replaced. Next occurrence of key string is found with (next).

- colors and character set Attributes

1 White	2 Green	3 Yellow	4 Cyan	5 Red	6 Blue	7 Magenta	8 Black
1 (next)	2 DRO	3 JoinLine	4 New File	5 Reverse	6 Set Attr	7 Mod:Color	8 Normal
1 INS/REP	2 INS Line	3 Cutchar	4 Cut Line	5 Paste	6 Save	7 Srch/Rep	8 Quit

Mod:Misc toggles to Mod:Color. CTRL f1-f4 show choices for foreground colors and CTRL f5-f7 some background colors. Set Attr selects choosen colors. Reverse reverses fore- and background. Normal (sf8) toggles Hex and Literal.

- Block function

1 Mark	2 Unmark	3 Blk Move	4 Blk Copy	5 Blk Del	6 Blk Save	7 Ins File	8 CS:ASCII
1 (next)	2 DRO	3 JoinLine	4 New File	5 Reverse	6 Set Attr	7 Mod:Misc	8 Normal
1 INS/REP	2 INS Line	3 Cutchar	4 Cut Line	5 Paste	6 Save	7 Srch/Rep	8 Quit

Block function keys allow manipulation of big text blocks

- File access

1 Mark	2 Unmark	3 Blk Move	4 Blk Copy	5 Blk Del	6 Blk Save	7 Ins File	8 CS:ASCII
1 (next)	2 DRO	3 JoinLine	4 New File	5 Reverse	6 Set Attr	7 Mod:Misc	8 Normal
1 INS/REP	2 INS Line	3 Cutchar	4 Cut Line	5 Paste	6 Save	7 Srch/Rep	8 Quit

With the marked function keys files may be read from and written to the selected disk.

- END or cancel

1 Mark	2 Unmark	3 Blk Move	4 Blk Copy	5 Blk Del	6 Blk Save	7 Ins File	8 CS:ASCII
1 (next)	2 DRO	3 JoinLine	4 New File	5 Reverse	6 Set Attr	7 Mod:Misc	8 Normal
1 INS/REP	2 INS Line	3 Cutchar	4 Cut Line	5 Paste	6 Save	7 Srch/Rep	8 Quit

To leave the editor use **Quit**. This key is useful also to cancel functions.

5.3.4. CURSOR KEYS

Like the function keys, there are three sets of cursor keys: Normal, shifted, control and shift control. These are used to provide quick cursor movement and color painting.

- Unshifted Mode



The cursor keys move the cursor in the direction of the arrow.

- Shift Mode



Moves the cursor and set the current color attribute(s.) [SHIFT] [F6])

- Control Mode



Moves the cursor to the next word right or left from actual cursor position.



Moves the screen (18 lines) up or down.

- Shift - Control - Mode



Moves the cursor to the blank after the last or rather the first edited character in this line.



Moves the cursor to the beginning of the file or to the end of the file.

- Home



The home - key moves the cursor to the place, where it has been before the last changing.

- Backspace



The Backspace - key deletes the last written character.

5.4. DESCRIPTION

5.4.1. CHARACTER KEYBOARD

The character keyboard including the tab, return and backspace keys, allows the user to manipulate the text accepted exactly like a normal typewriter. If, while entering text, the end of a line is reached, no more characters are allowed and an error message is displayed. The tab key positions the cursor up to 4 spaces up to the next tab position which is 5, 9, 13, e.g.

The action of all of these keys depends on the insert/replace key (Bottom Bank key f1). When in insert mode, the return key causes the insertion of a new line, and tab causes subsequent text on a line to be written over. When in replace mode, the return key replaces the current character with a new line.

Note that when the cursor is in the middle of a line of text, a backspace moves subsequent text over one space to the left and removes the characters previously in that space in insert mode. However, a backspace replaces the character by space and moves the cursor to the left in replace mode.

5.4.2. FUNCTION KEYS

In the following the function keys will be explained in detail.

5.4.2.1. EDITING KEYS



| INS / REP | - Insert / Replace

INS/REP changes between INSERT- and the REPLACE - Mode. The current state is highlighted in blue.



| INS LINE | - Insert Line

INS LINE inserts a line below the actual cursor position.

Note that insertion behind the cursor make the insertion at the beginning of a file impossible. With the PASTE - command the user can insert a line at the beginning of the file.

**| CUT CHAR | - Cut Character**

CUT CHAR removes the current character and puts it in the paste buffer (s. PASTE - |f5|). |BS| (Backspace) will remove the character left from the cursor position. |f3| and |BS| move the rest of the line one place left, independent, whether the Insert- or Replace mode is active.

**| CUT LINE | - Cut Line**

CUT LINE removes the current line and puts it into the paste buffer (s. |f5|). The cursor is moved to the line that replaces the cut line.

**| JOIN LINE | - Join Line**

JOIN LINE joins two consecutive lines together if the second line can fit in the space at the end of the first line. A space is used to separate the contents of the two lines and the cursor is placed at the first character of the second line.

**| PASTE | - Paste From Buffer**

PASTE puts the contents of the PASTE buffer (character, line or block) at the actual cursor position in the file. This allows lines to be moved or copied.

The paste of a line is above the cursor and the paste of a character is before the cursor. This allows the user to recover from erroneous use of the CUT keys with a single keystroke.

5.4.2.2. SEARCH- AND REPLACE KEYS



|SRCH / REP| – Search and Replace

The function key [f7] prompts the user for entry of a string terminated for example by a return. This string is then mapped with the text in the file according to the current character set selected. If the string is found, the cursor is moved to the start of that string, otherwise a message is displayed. The search will wrap around the bottom of the file until it reaches the current cursor position.

The search string can be selected (like the filename) with the keyboard cut and backspace keys when the user is being prompted. In addition, HEX and literal entries are allowed while entering or editing the search string. Don't cares can be specified by '?' in the search strip.

With QUIT [f8] the function can be canceled.

– Start Search Function

The search function can be started with four keys:



Starts the search function from the cursor position forward.



Starts the search function from the cursor position backwards.

– Replace Function

After entering the search string, the user can activate the replace function using . The user is prompted for the replace string and should be entered the same way as the search string RETURN. The replace will be started analogous to the search function.

Pressing the -key the user returns to the search function.

– Replace

Function-keys SHIFT-F1 to SHIFT-F3 become available for replacing (they are highlighted in magenta). Four different actions are possible :



|REPLACE ALL| – Replace all existing Strings.



|REPLACE| – Replace the great string with REPLACE-String and search for the next occurrence of SEARCH-String.



|NEXT| – Search the next string

After a search string has been entered, the NEXT key can be used to find the next occurrence of the string without re-entering the string.

Pushing another key, cancels the search / replace function.

5.4.2.3. ENTRY MODE KEY



[[NORMAL,HEX,LITERAL]] - Select Entry Mode

This key selects one of the three entry modes:
NORMAL, HEX and LITERAL.

Normal mode allows all ASCII characters to be entered except BACKSPACE, TAB, RETURN and CTRL-P. Entering CTRL-P will cause a Teletex CTRL-P with red background and white foreground entered into the file. This special character is used by the application software as a termination character for a long string for transmission.

In Hex mode, all characters are entered a nibble at a time. The only characters allowed for entry are the hex characters (either case); tab and carriage return are interpreted and expanded. Other characters will produce a beep. After a nibble is entered, it will appear beside the HEX prompt at the lower right of the screen. The cursor will jump to this point as well. Pressing backspace will clear the character and cause the cursor to return to the previous screen position. Entering a second nibble will cause the compiled character to be put on the screen and the cursor position restored.

Literal mode allows all ASCII characters to be entered, and the characters : BACKSPACE, TAB and CARRIAGE RETURN are not interpreted.

5.4.2.4. CHARACTER SETS



[CS:[ASCII, EBCDIC, HEX, TTX, JIS8]] - Select Character Set

If the editor is in **MOD: MISC** (switching between MODE=MISC and MODE=COLOR can be done pressing SHIFT -F7), the user is able to chose cyclical between character sets: ASCII, EBCDIC, HEX, TTX (Teletex) and JIS8 (Katakana). All characters entered have the current selected character set as part of their attributes.

5.4.2.5. COLOR EDITING KEYS



[MOD: [MISC,COLOR]] - Select Color

By default the editor is in **MOD: MISC**. Therefore the block functions are active. To modify the color the editor should be in **MOD:COLOR** .(Use [SHIFT] [f7])

The default color for entry of text is white foreground on a black background. To change the colors use the top line of the function keys. The first four keys changes the foreground and the last four the background. The current color combination is reflected by the color of the **SET ATTR** key middle line. **REVERSE** [SHIFT] [f6] inverts the foreground and background color.

When the desired color combination is chosen, select the **SET ATTR** (Middle Bank f6) key. Text will be entered in this color combination until the **END ATTR** (Middle Bank f6) key is selected at which time the editor returns to the default value.

The color of a further entered text can be changed while moving the cursor with the shifted cursor keys on top of the text.(see 5.3.4.).

5.4.2.6. BLOCK FUNCTION KEYS



|MARK| - Mark Block

MARK selects the beginning or the end of a block. A selected block is displayed inverse. At the same time some top line function keys are highlighted indicating that they become active now. The selected area can be enlarged or reduced.



|UNMARK| - Unmark Block

UNMARK cancel the selection.



|BLK MOVE| - Move the marked block

BLK MOVE moves the selected block towards the cursor position.

Method:

1. Mark the block ,
2. Move Cursor to target
3. Move block



|BLK COPY| - Copy the marked block

BLK COPY copies a selected block to the cursor position. If there's no more space in the editor buffer an error message appears.



|BLK DEL| - Delete the marked block

BLK DEL deletes a selected block. If a block has been deleted accidentally, it can be restored by using PASTE |f5|.



|BLK SAVE| - Save the marked block on disk

With BLK SAVE the selected block can be written to a file.
(see |SAVE|)

5.4.2.7. FILE ACCESS

**|SAVE| – Save the textbuffer on disk**

Using SAVE the user can write the file to the floppy or the hard disk. The user will be prompted for the filename.

Entering the filename, following functions are active:

CUT CHAR or BS	: delete a character ;
CUT LINE	: delete the whole filename;
PASTE	: old filename back;
QUIT	: cancel the function;
DR XX	: select drive.

Once the filename is entered, the file is removed if it already exists, and then created.

**|DR[0,1]| – Change drive**

DR[0,1] select the drive for reading or writing.*

**|NEW FILE| – Edit a new file**

Using NEW FILE a new file can be loaded from floppy or hard disk. The user has to enter a new filename. To restore the old filename he has to enter PASTE |f5|. Note that the old file is not saved automatically.

**|INS FILE| – Insert text from disk**

Using INS FILE a file can be inserted in the already existing text.

5.4.2.8. DECOMPRESS FILES

All files will be stored in a compressed format. With the FORTH – command

" filename" ?COMPRESS

The user gets the file format. In System prior MF1.5, BRI_1.5, PRI_1.5 (V3.2, MF1.2, BRI_1.2 e.g.) the OLD EDITOR file format is used. To convert the compressed file format to the uncompressed file format use.

" filename1" " filename2" DECOMPRESS_FILE .

This will automatically convert the compressed file 1 to the uncompressed file 2.

* If a hard disk is installed, this function key can be used to select the Winchester WD0, WD1 ...

5.4.2.9. EXIT



|QUIT| – Quit Editor / Abort a function

Pressing QUIT the user can leave the editor. Note that the file will not be saved automatically. To avoid leaving the editor accidentally without saving the EDITOR-Buffer contents the user is prompted for a confirmation before leaving the EDITOR. Only the inputs "Y", "y" or |f8| will be accepted before leaving the editor.

Note: If you leave the EDITOR by mistake restore your edited file typing EDITM !

6. ARCHITECTURE

The K1197 is based on an especially developed hard- and software architecture which consists of

- a VME bus based multiprocessor system with up to seven 32 bit microprocessors and up to 15 MByte RAM.
- various physical communication interfaces (from RS 232-C, V.36, V.35, X.21 to ISDN BAI and ISDN PRI).
- a real time, multitasking, message oriented multi processor operating system MF 2.1 (Multi Forth).
- easy to use, window oriented user interface.
- programmable in an enhanced version of FORTH, a high level language and development environment all in one. FORTH supports the principles of modern programming style such as structured programming, top down design, modularity and readability. It minimizes maintenance effort while producing fast and compact code. Up to seven concurrent FORTH machines are supported.
- different protocol specific analyzer and simulation/emulation packages with SDL oriented test language are available.

6.1. HARDWARE

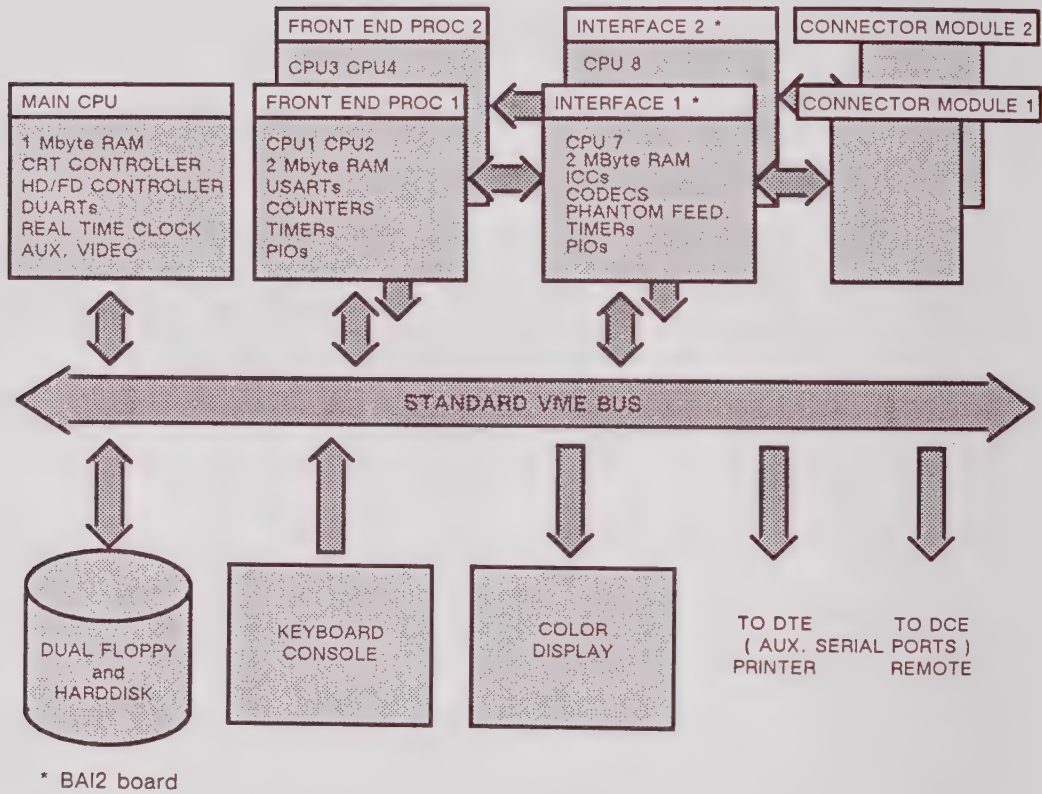


Figure 6.1-1: Hardware Architecture

6.2. MEMORY MAP

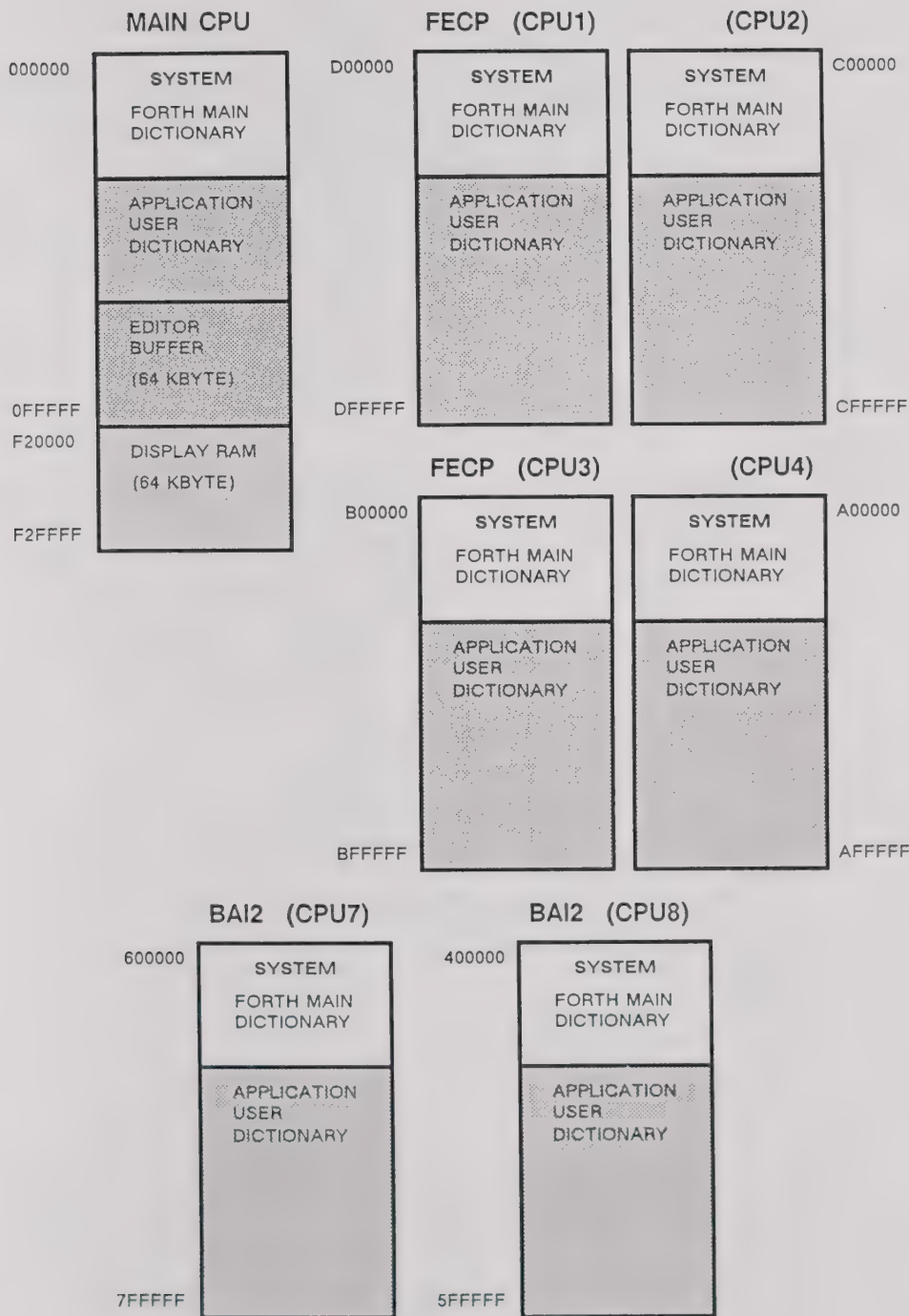


Figure 6.2-1: Memory Map

6.3. SOFTWARE

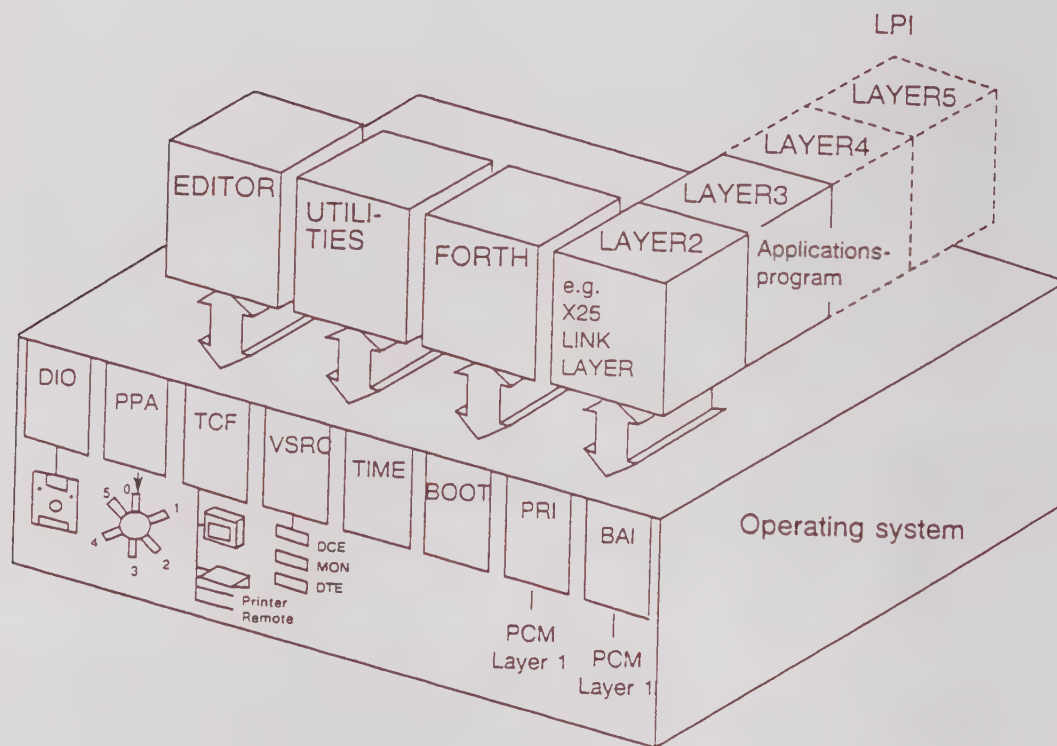


Figure 6.3-1: Software Architecture

DIO	Disk I/O
PPA	Processing Power Allocator = Process Scheduler
TCF	Terminal/Console Function
Time	Timer
PRI	Primary Rate Interface, Layer 1
BAI	Basic Access Interface, Layer 1

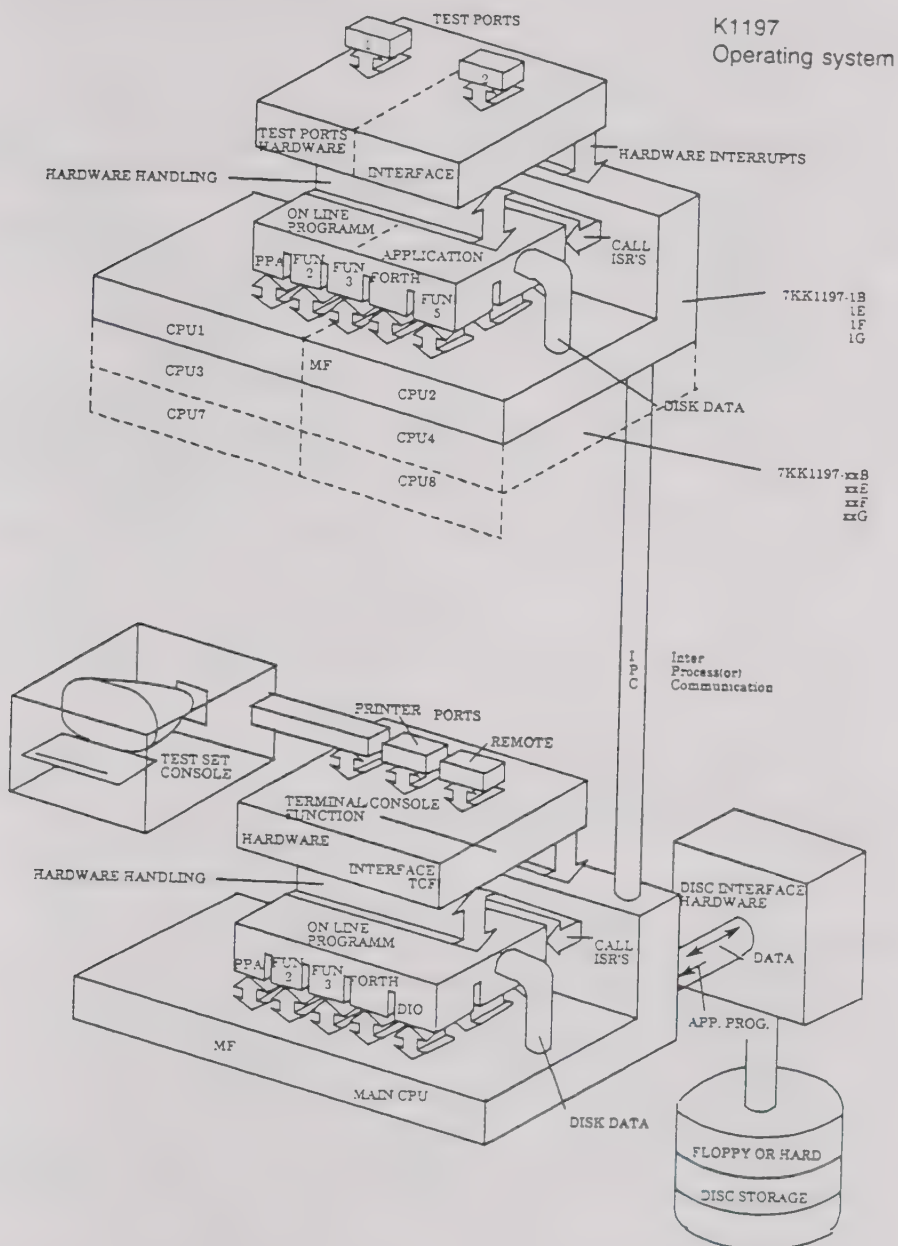


Figure 6.3-2: K1197 Operating System structure with V/X Interface

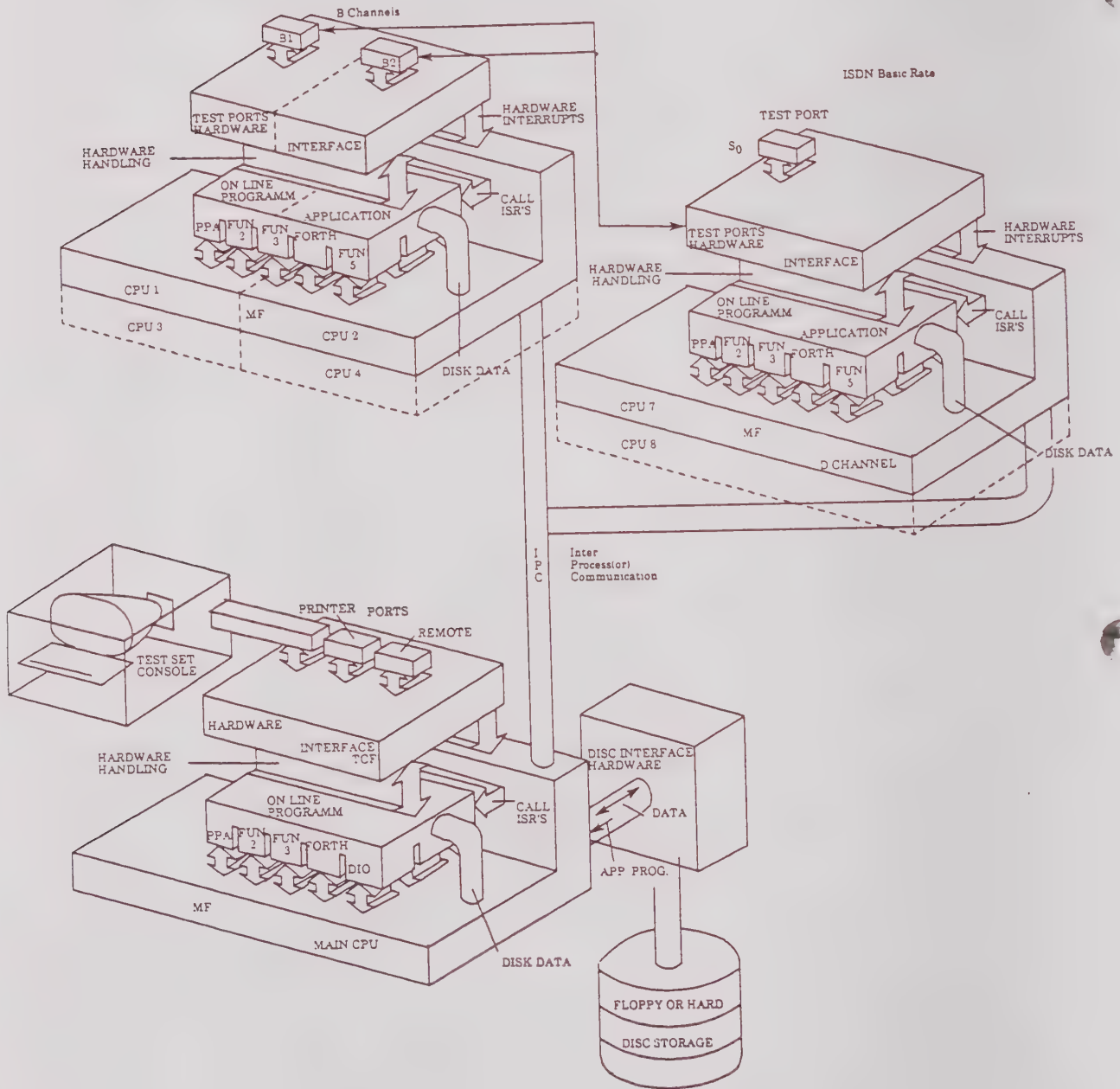


Figure 6.3-3: K1197 Operating System Structure with ISDN Basic Access

7. HANDLING OF THE K1197

7.1. HOUSING

The housing of the K1197 contains a alumine frame and two covers on top and at the bottom. In use of the K1197 it could be necessary to open the top housing cover.

Proceed as follows:

- 1.) Switch off the K1197
- 2.) **Disconnect mains plug from the device**
- 3.) Remove housing cover (four screws)

7.2. MONITOR

The surface of the monitor screen is covered with a special anti-reflection layer. Use only smooth anti-static materials to wipe off the dust.



DANGER

After removal of the housing cover take care of the following.

The monitor tube and its electronic parts use high voltages for its operation.

Some parts of the monitor tube and its electronics are keeping its high voltages for hours although the K1197 is disconnected from mains.

Do not touch these parts !

On the back of the K1197 housing two control elements of the monitor can be found.

- 1.) The intensity can be controlled by turning the INTENSITY-knob, a mid position is marked.
- 2.) In case of a permanent discolourings on the monitor screen use the DEGAUSS-knob. The best results can be achieved by pushing the knob, waiting for at least five seconds (to give the demagnetizing-coil time to load up) and then repeating the procedure as long as necessary.



CAUTION

Although floppy disks which are inserted in the drives will not be affected by the demagnetizing field of the degauss function, take care, that no floppy disks are in a distances less than 30 cm from the screen, when activating this function.

7.3. POWER SUPPLY

The power supply contains a mains fuse which can be replaced from the back of the K1197 by using a screw driver.



DANGER



After removal of the housing cover take care of the following.

The power supply uses high voltages for its operation.

Some parts of the power supply are keeping its high voltages for hours although the K1197 is disconnected from mains.

Do not touch these parts !

7.4. HARD DISK DRIVE

An access to the hard disk drive will be displayed by a LED on front of the drive.

For highest reliability in data storage the hard disk needs 20 minutes to warm-up.

An auto-retrack function will park the drives heads in transport position after power down.

7.5. FLOPPY DISK DRIVES

An access to one of the floppy disk drives will be displayed by a LED on the front of the drive.



CAUTION

Keep away all dusty or greasy materials from the floppy disks and the floppy disk drives. For highest reliability it is recommended to check the drives annually by the service.

7.6. BOARDS



CAUTION

When handling the boards the regulations for electrostatic devices (ESD) must be observed.

Do not use any K1195 boards in K1197, damages can occur !

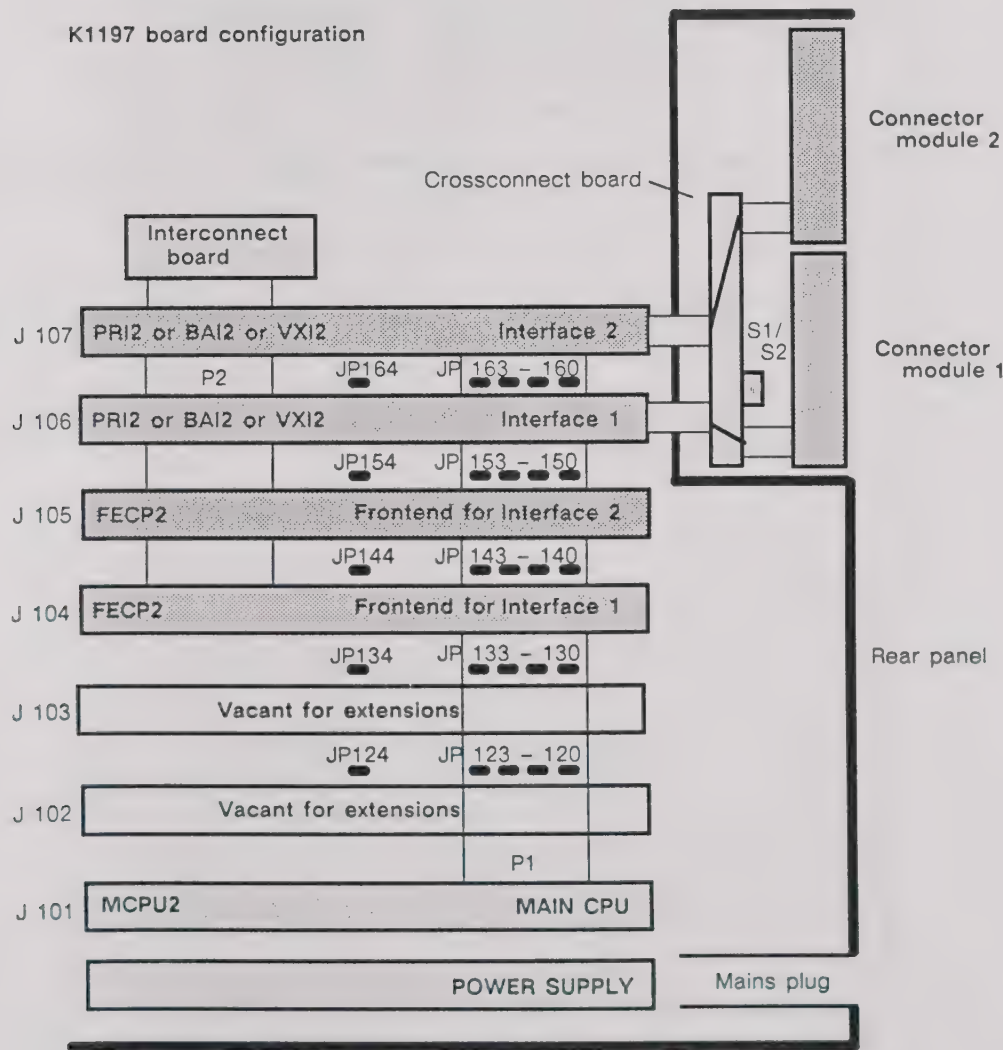


Figure 7.6-1: K1197 board configuration

7.6.1. BACKPLANE

The Backplane contains the jumpers 120–124, 130–134, 140–144, 150–154 and 160–164. They have the following functions:

- 1x0: Bus grant_0_in to Bus grant_0_out
- 1x1: Bus grant_1_in to Bus grant_1_out
- 1x2: Bus grant_2_in to Bus grant_2_out
- 1x3: Bus grant_3_in to Bus grant_3_out
- 1x4: Interrupt acknowledge_in to Interrupt acknowledge_out

The configuration of jumpers is as follows:

- J101: No jumpers
- J102: Jumpers 120–123 and 124 always present
- J103: Jumpers 130–133 and 134 always present
- J104: Jumpers 140–143 needed if no FECP2 for interface 1 is inserted
and J144 always present
- J105: Jumpers 150–153 needed if no FECP2 for interface 2 is inserted
and J154 always present
- J106: Jumpers 160–163 needed if no BAI2 for interface 1 is inserted
and J164 always present

Note: There are some extra jumpers in the small plastic bag on top of the drives housing.

7.6.2. MAIN CPU

A Real Time Clock (ICM 7170) is included on the Main CPU board. This RTC is battery backed up. Since the chip has low power detection circuitry, no external circuitry is required to support the battery.

With the used battery the unit will keep time for at least 3 years. The battery is soldered in Pos. B401 and should be replaced only by experienced personnel.

Proceed as follows to replace the batteries :

1. Remove upper housing cover (see 7.1)
2. Pull out Main CPU board from Pos. J101 of the backplane
3. **Remove Jumper Pos. J401 on Main CPU**
4. Remove battery from board Pos. B401 (soldered)
5. Replace Battery, use one of the following:

VARTA CR 1/3 N SLF Ord. Nr. 6131201501
SANYO CR-1/3N-FT1

General data of battery:

Voltage 3 V

Capacity 160 mAh

Self discharge at 20 °C < 1% p. a.

Maximum temperature 60 °C

(Operating and Non-operating)

6. Insert Jumper at Pos. J401 on Main CPU
7. Insert Main CPU board on Pos. J101 of the Backplane
8. Screw back housing cover
- 9 Reconnect mains plug and switch device on
10. Set time and date

7.6.3. FRONT END CPU (FECp)

The FECp is used to send and receive serial data streams with four separate MPCCs (Multi Protocol Communication Controllers) and two separate CPUs (68000) with one MByte RAM each. The K1197 contains at least one FECp in slot J104, if two interfaces are used a second one must be installed in slot J105. Those two FECps use different addresses which can be selected by the jumpers J301 and J302. The position of the jumpers are shown below.

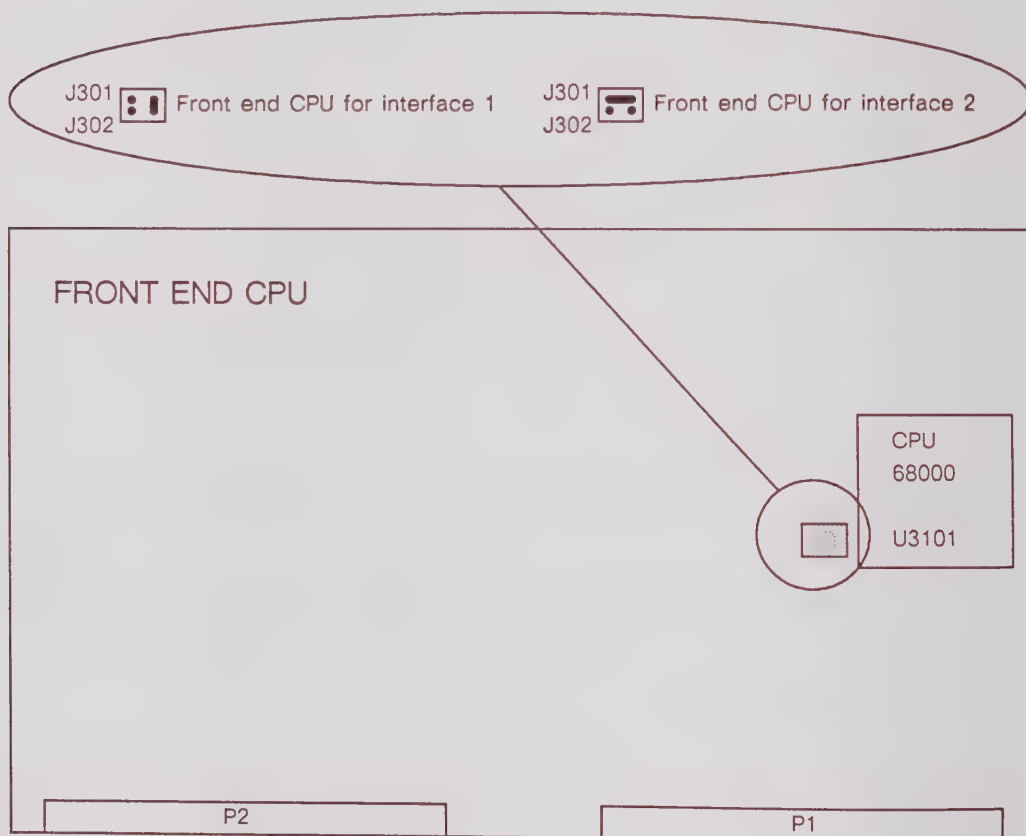


Figure 7.6.3-1: FECp2 – Parts side of the board

All other jumpers should not be changed by the user !

7.6.4. INTERFACE BOARDS AND CONNECTOR MODULES



CAUTION

Before installation of a connector module check if the combination is correct

BAI2 is used with:

S_o – Connector module

U_{po} – Connector module

PRI2 is used with:

CEPT – Connector module (2,048 MBit/s)

T1 – Connector module (1,544 MBit/s)

Proceed as follows to replace the interface board(s) :

1. Remove upper housing cover (see 7.1)
2. Remove connector modules (4 screws each) and crossconnect board (4 screws) from the back of K1197
3. Pull out interface board(s)
4. Insert new interface board(s)
5. Install crossconnect board (see 7.6) and the correct connector modules
6. Screw back housing cover
7. Reconnect mains plug and switch device on

The interface boards PRI2 and BAI2 use different addresses depending on their slot positions (J106 or J107).

These have to be configured **on the interface board** as follows:

	Interface 1 (J106)	Interface 2 (J107)
PRI2	J1101 inserted	J1102 inserted
BAI2	JP101 free	JP101 inserted

The interface board VXI2 can be used in both slot positions (J106 or J107) without any change.

Note: By using a interface board PRI2 single or in combination with other interface boards, the interface board PRI2 has to be used in the slot position J106

7.6.5. CROSSCONNECT BOARD

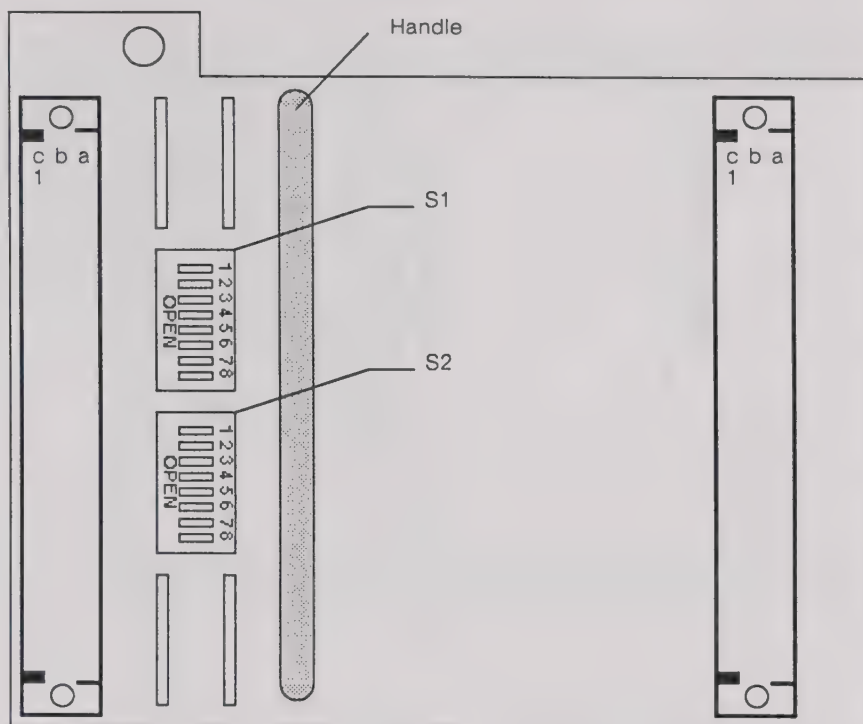


Figure 7.6.5-1: Rear view of crossconnect board

The crossconnect board serves as a unit for the electrical and mechanical connection of the connector modules with its interface boards and the electrical connection between the interface boards.

For data transfer between the interface boards it is necessary to synchronize the clocks. One interface board acts as the clock master the other as slave. For the combination PRI2–BAI2 the PRI2 is always the master, so the BAI2 has to be programmed as slave. If two BAI2 are used the user can select which board should be the clock master.

Data transfer is controlled by two 8-switch elements S1 and S2 which are accessible if the left connector module is removed.



CAUTION

If only one interface board is installed all switches should be OPEN.
If two interface boards are used all switches should be **not** OPEN.

Do not use K1195 interface boards in K1197 this can damage the boards.

8. AUXILIARY INTERFACE

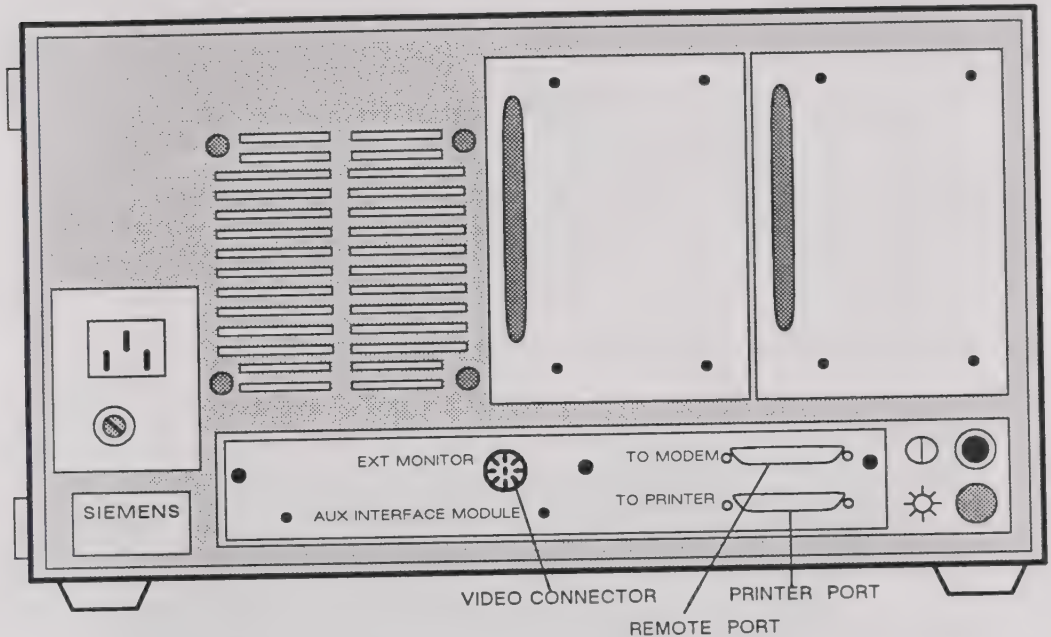


Figure 8-1: Rear view with Auxiliary Interface

8.1. REMOTE PORT

Remote Control of K1197 applications can be done using the TO MODEM connector on the auxiliary interface module on the back panel of the Protocol Tester.

The remote control port is configured to use async, ASCII, 8 bits, no parity, 1200 Baud. The K1197 does not echo received characters, if local display is wanted on a terminal, the terminal should be set to half duplex mode.

Commands on the remote control line are to be entered followed by a carriage return character. The program will process the input, and send the command line display to the remote site followed by the prompt " ROK " and the carriage return and the linefeed characters. The programs will process the remote commands the same way as if they had been entered on the local keyboard.

8.1.1. REMOTE SPEED SETTING

The speed of the remote port can be changed with the command **RSPEED**. It is used in the format: 1200 **RSPEED**, where 1200 is the line speed wanted. Speeds of 110, 150, 300, 1200, 2400, 4800 and 9600 are supported. The speed setting is stored in the FORTH variable R-SPEED and is used to configure the speed setting when programs are started.

8.1.2. PINOUT

TO DCE, MALE CONNECTOR

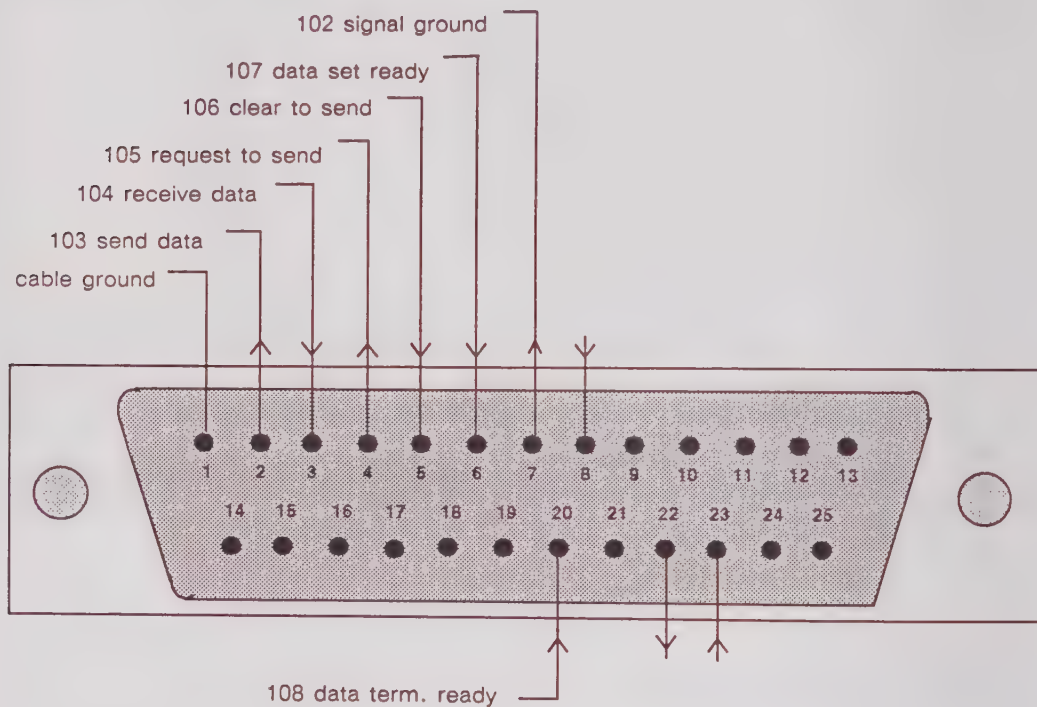


Figure 8.1.2-1: To DCE, Male Connector

All signals meet CCITT V.28 (RS232-C) specifications
($|V_{out}| > 5\text{volts}$, Receiver thresholds $\pm 3\text{volts}$)

8.1.3. REMOTE SCREEN DISPLAY

The K1197 screen display can be sent to the remote control port by entering in the following commands.

RSCREEN	(--) The 23-line K1197 screen display is sent to the remote control port.
N RLINE	(N --) The specified line (N, $0 \leq N \leq 22$) of the K1197 screen is sent to the remote control port, eg. 0 RLINE will send the top line of the screen display to the remote control port.
RCR	(--) Carriage return and linefeed characters will be sent on the remote control port.
RFLINE	(--) The function key line of the display (bottom line) will be sent to the remote control port.
RCLINE	(--) The command lines of the display will be sent to the remote control port.
RSLINE	(--) The status lines (top 3 lines) of the display will be sent to the remote control port.
RMT_DIR	(--) The disk directory in short format will be sent to the remote control port.
RMT_DIRL	(--) The disk directory in long format will be sent to the remote control port.

8.1.4. REMOTE/LOCAL COMMANDS

Some FORTH words have been redefined to give them different operations when remote control is being used compared to local control. The variable MBOX-1 is used to indicate that remote control command entry has been done and will contain the value of the constant **RMT_STR_IND** when remote commands have been entered and will contain the value of the constant **STR_IND** when local keyboard control is being done. The following description of these FORTH words explains the difference in operation when using remote control.

CR	- CR will execute the display of a newline in the command display window and will send a display of the previous line to the remote control port.
WCR	- WCR will execute the display of a newline in the currently open display window and will send a display of the previous line to the remote control port if RMT_ON is selected.
EDIT	- EDIT cannot be called when using remote control.
COLD	- COLD cannot be called when using remote control.
EXECB	- EXECB cannot be called when using remote control.

8.1.5. REMOTE FUNCTION KEY ENTRY

Entry of a cursor key can be simulated by the following commands:

CU!	- Simulates the cursor-up key pressed
CD!	- Simulates the cursor-down key pressed
CL!	- Simulates the cursor-left key pressed
CR!	- Simulates the cursor-right key pressed
CSU!	- Simulates the cursor-shift-up key pressed
CSD!	- Simulates the cursor-shift-down key pressed
CSL!	- Simulates the cursor-shift-left key pressed
CSR!	- Simulates the cursor-shift-right key pressed
CCU!	- Simulates the cursor-control-up key pressed
CCD!	- Simulates the cursor-control-down key pressed
CCL!	- Simulates the cursor-control-left key pressed
CCR!	- Simulates the cursor-control-right key pressed
CSCU!	- Simulates the cursor-shift-control-up key pressed
CSCD!	- Simulates the cursor-shift-control-down key pressed
CSCL!	- Simulates the cursor-shift-control-left key pressed
CSCR!	- Simulates the cursor-shift-control-right key pressed

The remote entry of function / cursor keys is used for example to control the Menu-System or in application programmes with Test Manager scenarios running where function / cursor key entry is expected.

8.1.6. REMOTE PLAYBACK OF DATA FILES OR CAPTURE RAM

Application packages that provide playback of data files or capture ram can be controlled with the following commands. Note that result reporting is displayed correctly when scrolling in the forward direction (TOP, F, FF, TRANSFER) but the reports appear reversed when scrolling is done in the backward direction (BOTTOM, B, BB). In addition to the commands listed here, reference should be made to the Users Manual of the application package being used for the complete list of commands to do capture, recording, playback, and results formatting.

RMT_ON	- Playback reports will be sent to the remote control port (except CCS No. 7 packages). This setting is indicated by the value 1 in the variable REMOTE-ACTIVE. Also, display lines that are terminated with a WCR will be sent to the remote control port.
RMT_OFF	- Playback reports are not sent to the remote control port. This setting is indicated by the value 0 in the variable REMOTE-ACTIVE.
PLAYBACK filename	- The disk data file with the title filename will be opened for playback.
TOP	- The first screen (14 lines) of the playback will be reported.
BOTTOM	- The last screen (14 lines) of the playback will be reported.
F	- The next line of the playback will be reported.
B	- The previous line of the playback will be reported.
FF	- The next screen (14 lines) of the playback will be reported.
BB	- The previous screen of the playback will be reported.
TRANSFER	- The entire disk file or capture ram playback will be reported.
QUIT_TRA	- A transfer of data playback can be terminated at any time with the QUIT_TRA command.

8.1.7. TEST MANAGER

The first test recommended is the connection of a terminal.

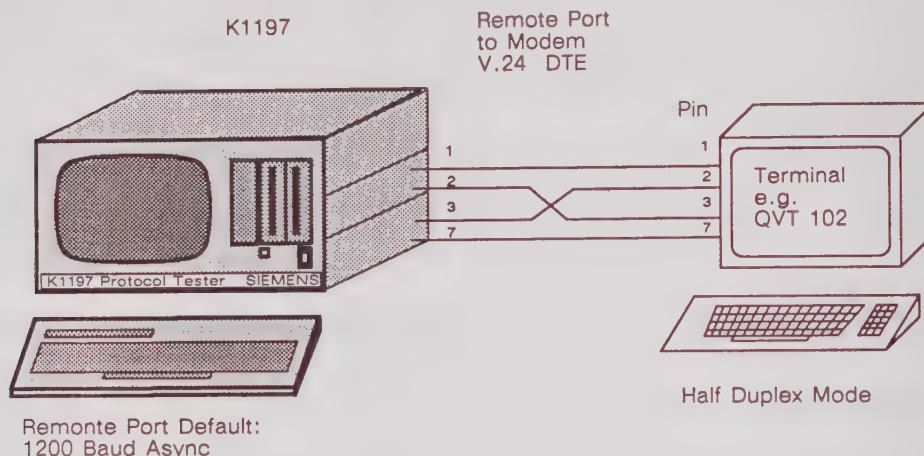


Figure 8.1.7-1: Connecting a terminal to the remote port

1. Remote Operation

Sending data to the terminal:

"Hello terminal" COUNT 1 REMOTE_OUT

RCR

(or REMOTE_OUT_W, which will wait until the last byte has been sent)
(sends a CR)

Terminal Operation

If the RETURN-key is pressed, the K1197 responds with ROK-X (X is the number of the selected CPU).

(For ISDN-D-Channel, RMT_ON must be set before this reaction is activated).

Key in RSCREEN: This sends the current K1197 screen display to the terminal. For more information, see K1197 Forth Manual.

Receiving Data with the K1197

Commands can be entered from a terminal similar to local command entry, but line editing functions are not provided. They are terminated by the return key. They are then interpreted and the result is displayed in the command window.

2. Test Manager

Keywords to recognise a received remote string.

Please type in

In the case of ISDN-D-channel:

```

: ?RMT_STR                                (-- YES or NO)
      EVENT-TYPE @ STRING-ENTRY#  =

      IF                                (is a string received?)
          MBOX-1 @ RMT_STR_IND  =
                                (was it a remote string?)
      ELSE
          NO
      THEN
;

```

In the case of X.25:

```

: ?RMT_STR                                (-- YES or NO)
      EVENT-TYPE @ RMT_STR_IND  =
                                (was it a remote string?)
;

```

Application example for the Test Manager:

```

0 STATE{  ?RMT_STR                                (was it a remote string?)
          ACTION{
              ."-->" RMTGETS TYPE
                                (display string on screen)
          }ACTION
        }STATE

```

The routine RMTGETS leaves the address and the length of the string received on the stack.

8.2. PRINTER PORT

Data output to a printer can be done using the TO PRINTER connector on the auxiliary interface module on the back panel of the Protocol Tester.

The printer port configuration is defaulted for: Async., 1200 Baud, 8 Bits/character, ASCII, DTR and XON/OFF control, LFCR end of line, 80 characters/line, 60 lines/ page.

8.2.1. PINOUT

TO DTE, FEMALE CONNECTOR

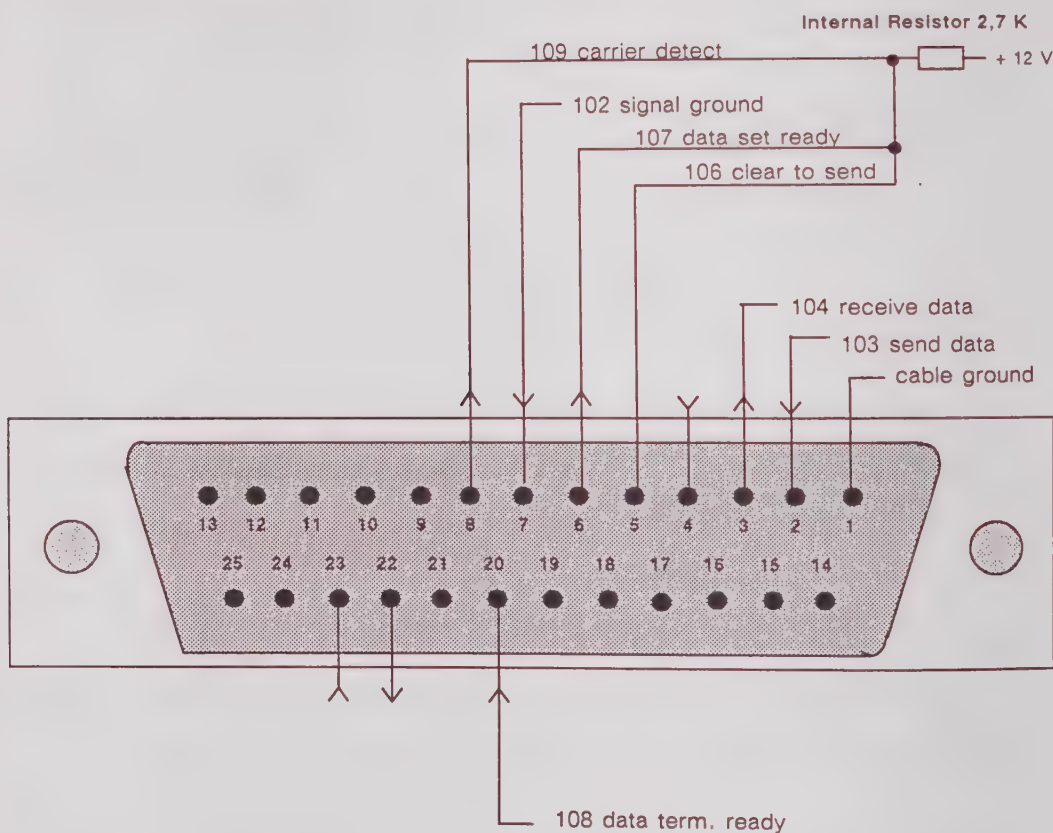


Figure 8.2.1-1: To DTE, Female Connector

All signals meet CCITT V.28 (RS232-C) specifications
 ($|V_{out}| > 5$ Volts, Receiver thresholds ± 3 Volts)

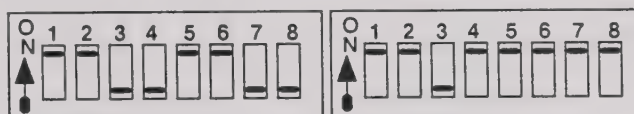
8.2.2. CONNECTING A PRINTER SIEMENS PT88S

Printer Type: PT88S-32 (..PD 2 or something similar) or -34 (EPSON-compatible) with Serial Interface SAP S22767-B1-A100 (must be ordered separately).

Printer-cable: Normal V.24 cable with two 25-pol. male connectors

Standard configuration of PT88S for K1197

Code switches on printer front under the front.



For PT88S-34 the switch No.3 must be on

The rear code switches onboard of the serial interface

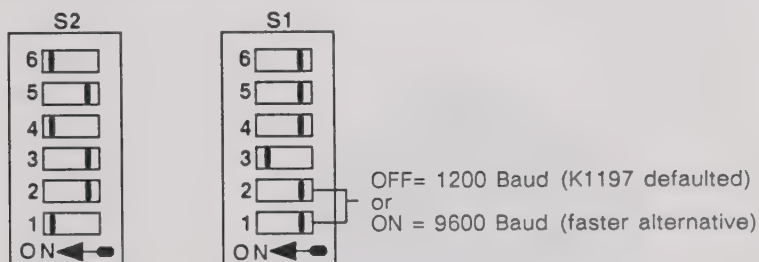


Figure 8.2.2-1: Settings of code switches of PT88S

If you wish to test the printer, the following key combination on the K1197 is recommended.



Print a hardcopy of the current screen display. This function generates no K1197 lookup, even when printer fails or cable faulty.

8.3. VIDEO CONNECTOR

This port is used to connect an external monitor to the K1197.

8.3.1. PINOUT

All voltages with TTL levels, RGB signals in positive logic and SYNC signals in negative logic.

PIN	USE	Recommended connections for a IBM 8 PIN PLUG
1	Ground	6,5
2	Ground	
3	Ground	
4	Horizontal Sync.	7
5	Vertical Sync.	8
6	Red	2
7	Green	3
8	Blue	4

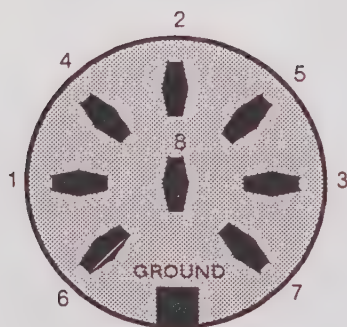


Figure 8.3.1-1: Female Connector DIN 45326

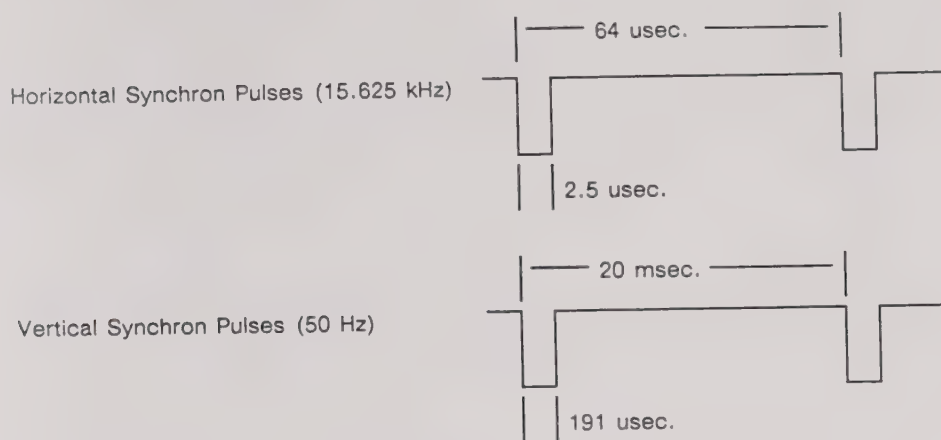


Figure 8.3.1-2: Timing diagrams

FORTH PROGRAMMING USER MANUAL

FOR DUAL INTERFACE PROTOCOL TESTER

SIEMENS AG

System Version : MF 2.2

Revision Date : March, 1992

1. WHY FORTH AND WHAT IS IT?	1-1
2. INTRODUCTION TO THE SIEMENS IMPLEMENTATION OF FORTH	2-1
3. GENERAL FORTH	3-1
3.1. INTRODUCTION	3-1
3.2. GENERAL FORTH WORDS	3-3
4. K1197 FORTH ENHANCEMENTS	4-1
4.1. INTRODUCTION	4-1
4.2. SYNOPSIS	4-1
4.2.1. COMMAND SUMMARY	4-2
4.3. GENERAL DEFINITIONS	4-3
4.4. MISCELLANEOUS COMMANDS	4-5
4.5. CASE CONSTRUCT	4-9
4.6. MAKE AND DOER CONSTRUCT	4-11
4.7. CONDITIONAL COMPILE OPERATIONS	4-11
4.8. INTERFACE LEAD CONTROL AND CHECKING	4-13
4.8.1. V.24 INTERFACE RS232C	4-14
4.8.2. V.11 INTERFACE	4-15
4.8.3. V.35 INTERFACE (34 PINS)	4-16
4.8.4. V.36 INTERFACE	4-17
4.8.4.1. 37 PIN CONNECTOR	4-17
4.9. BEEPER CONTROL	4-18
4.10. TIMER	4-18
4.11. TIME	4-19
4.12. DISK INTERFACE	4-21
4.12.1. DISK DRIVE SELECTION	4-21
4.12.2. FILE ACCESS CODES	4-21
4.12.3. FILE CLOSE CODES	4-22
4.12.4. DISK FILENAMES	4-22
4.12.5. FILE DESCRIPTOR	4-22
4.12.6. HARD DISK SUPPORT	4-27
4.12.6.1. ORGANIZATION OF A WINCHESTER DISK	4-27
4.12.6.2. COMMANDS	4-27
4.12.7. DATA RECORDING CONTROL	4-29
4.12.8. DISK PLAYBACK CONTROL	4-29
4.13. DISPLAY MANAGER	4-30

4.14. DISPLAY	4-35
4.15. M. DISPLAY ROUTINES	4-37
4.16. CURSOR POSITIONING	4-37
4.17. FUNCTION KEY MANAGER	4-38
4.18. KEYBOARD INPUT	4-42
4.19. ERROR RECOVERY & ERROR PROTECTION	4-42
4.19.1. SET_RETURN	4-43
4.19.2. SHELL{ AND }SHELL	4-43
4.20. CAPTURE BUFFER CONTROL	4-44
4.21. PRINTER PORT	4-45
4.22. REMOTE MODEM PORT	4-46
4.23. CONFIGURATION	4-47
4.23.1. INTERFACE	4-47
4.23.2. CLOCKING SELECTION	4-49
4.23.3. USART SETUP	4-50
4.23.4. ASYNC PORT CONTROL	4-52
4.23.5. CONFIGURATION BUFFERS FOR CONFIGURATION INFORMATION	4-54
4.24. CONFIGURATION OF ISDN INTERFACES	4-58
4.24.1. CONFIGURATION OF THE BASIC ACCESS INTERFACE (BAI) ..	4-58
4.24.1.1. INTRODUCTION	4-58
4.24.1.2. CONFIGURATION MODE RELATED COMMANDS	4-58
4.24.1.3. LAYER1 AND LAYER2 RELATED COMMANDS	4-60
4.24.1.4. B CHANNELS RELATED COMMANDS	4-61
4.24.1.5. BAI FORTH CONSTANTS	4-64
4.24.2. CONFIGURATION OF THE PRIMARY RATE INTERFACE	4-65
4.24.2.2. GLOBAL CONFIGURATION ROUTINES	4-66
4.24.2.3. ROUTINES FOR CHANNEL CONFIGURATION	4-68
4.24.2.4. CONFIGURATION AND MONITORING OF TIMESLOT 0	4-69
4.24.2.5. PRI CHANNEL ROUTING	4-71
4.24.2.6. PRI ERROR / ALARM GENERATION AND REPORTING ROUTINES	4-72
4.24.2.7. PRI INDICATION BUFFER EXTRACTION	4-75
4.24.2.8. PRI FORTH CONSTANTS	4-75
4.24.2.9. FURTHER COMMANDS	4-77
4.24.2.10. STRUCTURE OF THE PRI CONFIGURATION BUFFERS	4-78
4.24.2.11. STRUCTURE OF THE PRI INDICATION BUFFER	4-80
4.25. DATA ORIENTED	4-81
4.26. STATE MACHINE RELATED	4-83

4.26.1. EVENT QUEUE MANAGEMENT	4-83
4.26.2. KEYBOARD/PRINTER/REMOTE ENTRIES	4-84
4.26.3. IPC BUFFER EXTRACTION	4-85
4.26.4. MAILING	4-88
4.26.5. KEY - CONSTANTS	4-91
4.27. CODE CONVERSION	4-92
4.28. CRC CALCULATION	4-92
5. K1197 MULTIFORTH ENHANCEMENTS	5-1
5.1. K1197 INTERPROCESS COMMUNICATION WORDS	5-2
6. PROGRAM DEBUG TOOLS	6-1
7. FORTH ERROR MESSAGES	7-1
7.1. FORTH EXECUTION/COMPILATION ERROR MESSAGES	7-1
7.2. DISK OPERATION ERROR MESSAGES	7-2
7.3. K1197 ERROR MESSAGES	7-2
8. TEST MANAGER	8-1
8.1. OVERVIEW	8-1
9. DESIGN NOTES	9-1
9.1. KEYBOARD MANAGEMENT	9-1
9.2. HANDLING KEYBOARD INPUT STRINGS UNDER PROGRAM CONTROL:	9-1
9.3. DISPLAY ASPECTS	9-2
9.4. DESIGNING A CONTROL LOOP	9-6
9.5. PORT CONFIGURATION	9-7
9.6. TRANSMISSION ASPECTS	9-8
9.7. DATA RECEPTION ASPECTS	9-9
10. SIEMENS FORTH WORDLIST AND INDEX	10-1

.....

1. WHY FORTH AND WHAT IS IT?

The question "Why did Siemens choose FORTH when there are so many other programming languages available?" cannot be answered easily because there are multiple reasons. Siemens compared the advantages and disadvantages of the following high level languages:

BASIC
PASCAL
C
PL/M
APL
FORTH

There were five areas of specific interest:

EASE OF INTERACTION	- during development, operating and maintenance phases
EXTENSIBILITY TIME	- the ability to define new variables, functions, capabilities, etc.
EXECUTION TIME	- during compiling and operation
STRUCTURE	- structured programming with capability for top-down design and coding
COMPACT CODE	- with regard to size and speed

Although some of the programming languages fared well in particular areas, only FORTH excelled in all areas. FORTH is similar to LISP and APL in the area of interaction, it's hardware interface capabilities are similar to those of C and ASSEMBLER, and the high level nature is similar to COBOL, FORTRAN and PASCAL. In terms of speed FORTH is somewhere between BASIC and ASSEMBLER - approximately an order of magnitude faster than BASIC. Where speed is of utmost importance, words (commands) can be coded in assembly language.

FORTH is a universally accepted language, and the FORTH model belongs to the public domain. Many text books are available on FORTH. One of the more popular ones is:

STARTING FORTH by LEO BRODIE
published by PRENTICE-HALL, INC.

An excellent pocket guide is:

POCKET GUIDE to FORTH by LINDA BAKER and MITCH DERICK
published by ADDISON-WESLEY PUBLISHING CO. INC.

A brief description of the components of FORTH follows:

- WORD - a defined command or an executable procedure. A word may be defined as a group of already defined words. Information is usually passed to a word using reverse polish notation (RPN) on the stack.
- DICTIONARY - a threaded list of words which have been compiled. Once a word has been entered in the dictionary, it can be executed interpretively.
- STACK - there are two LIFO stacks: a parameter (computation) stack and a return stack. the parameter stack is used to transfer information between words on a post-fixed basis (i.e. RPN). The return stack is used for nesting of addresses and data.
- INTERPRETER - there are two interpreters in FORTH: an outer interpreter and an inner interpreter. The outer interpreter is the text interpreter which interprets and attempts to execute the commands entered. The inner interpreter is an address interpreter i.e. it executes the commands whose addresses have been previously compiled into the dictionary (like a p-code interpreter).

FORTH is actually many things:

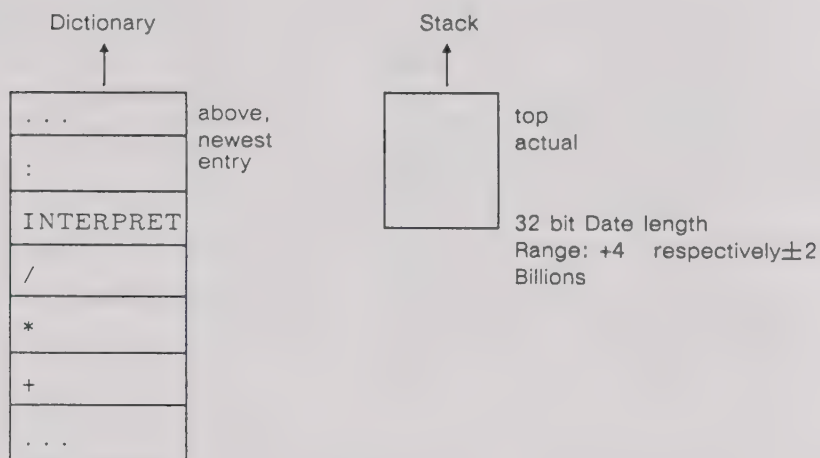
- a high-level language
- an assembly language (where necessary)
- an operating system
- a set of development tools
- a design philosophy (top-down)

Programming language Forth 1. view

FORTH was introduced in the 70th years by Charles Moore.

Structure: Indirect threaded language

Part of the implementation:



Interpreter: Stack-oriented, RPN (Reverse Polish Notation)

Compiler: Incremental-Compiler with linker

Adding of two numbers

<i>Example:</i>	<i>Interpreter</i>		
Typed:	1 2 + .	Answer:	3
Typed:	." Hallo"	Answer:	Hallo

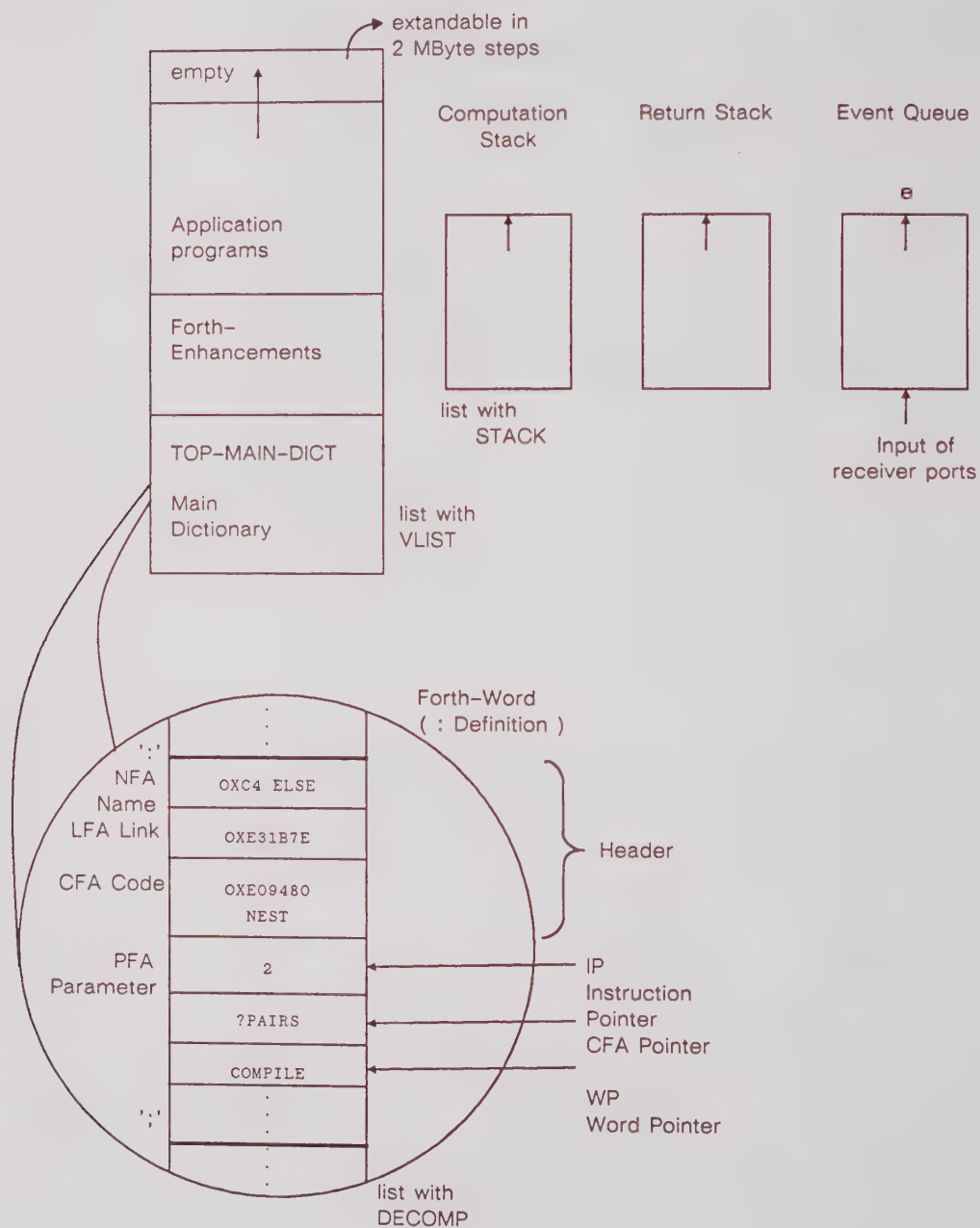
Compiler:

Typed:	: ADD ." Sum is" + . ;	Answer:	OK
Typed:	3 4 ADD	Answer:	Sum is 7

Time of Answer: ca. 15 Mikrosekunden each easy command

Forth enviroment 2. view Details

Dictionary



2. INTRODUCTION TO THE SIEMENS IMPLEMENTATION OF FORTH

Siemens has implemented an extensive FORTH vocabulary with some words taken from FORTH-79, Fig-FORTH, 68K FORTH as well as some extensions, eg. CASE construct. The basic vocabulary has been enhanced considerably in the following areas:

DISK FILE SYSTEM TOOLS	- to handle variable-length named files
EDITOR	- screen editor to generate variable-length named files
DISPLAY ENHANCEMENTS	- to take advantage of the color screen, multiple character sets, windowing and generally higher speed operation than is possible with traditional output commands. Control of two asynchronous ports is also provided.
TERMINAL INPUT ENHANCEMENTS	- to provide function key capability and input from remote terminals.
DATA COMMUNICATIONS CAPABILITY	- for transmission and reception of serial data, timer manipulation and a configuration capability to allow each test port to be individually configured so that true multi-port operation is possible.

FORTH uses an INPUT/OUTPUT interface (Called the FORTH front end or simply the Front End) to communicate with the RTE operating system. The FORTH front end handles the data communications interface as well as I/O to the screen and keyboard, printer and modem.

The FORTH Front End also allows a FORTH program to handle asynchronous events, generated by RTE such as data from the communications ports or keyboard input. This is called event mode.

In event mode, indications of asynchronous events are put into a fifo queue which can be accessed by a FORTH program.

Siemens has enhanced the **LOAD** screen concept significantly. The traditional FORTH **LOAD** concept is still supported with the command **EXEC**, except that a variable-length named file is used rather than a screen number. The significant enhancement is that the user dictionary, in binary form, may be saved or loaded and automatically started with the commands **SAVEB**, **EXECB** and **RUN_PROGRAM**.

3. GENERAL FORTH

3.1. INTRODUCTION

This section contains all the word definitions in Release 1 of fig-FORTH as well as Siemens extensions to the language which are of a general nature. The definitions are presented in the order of their ASCII sort.

LEGEND:

Types

W	Word
V	VARIABLE
C	CONSTANT
A	Address
(c)	like a constant

Source

fig	FORTH Interest Group
stand	standard, fig-forth79
siem	SIEMENS
siem-sys	SIEMENS system, not recommended for general use

Format

Parameter stack description follows the format:
 (value1/value2 -- value3)
 where '/' should be read as 'under' and ' -- ' as 'leaves'. The stack can be visualized as growing from bottom to top, with the last entry being on top.

Symbols

addr	memory address
b	8 bit byte (i.e. high 24 bits zero)
c	7 bit ASCII character
f	boolean flag. 0 = false, non-zero = true
ff	boolean false flag = 0
n	16 bit signed integer number
u	32 bit unsigned integer
tf	boolean true flag = non-zero
d	32 bit signed integer number
/	and
	or
{ }	optional

Note: All FORTH words are terminated by a SPACE character.

The description of FIG-FORTH words have been made available by:

The FORTH Interest Group,
P.O. Box 1105
San Carlos, CA 94070

Unless otherwise noted,

- all references to numbers are for 32 bit signed integers. The Siemens version of FORTH uses 32 bit stack entries.
- all references to display information refer to the FORTH window

3.2. GENERAL FORTH WORDS

!	(d/addr --) Stores 32 bits of d at address. Pronounced 'store'.
!CSP	(--) Saves the stack position in CSP. Used as part of the compiler security.
"	(-- addr) Character string input. Accepts character string entry up to a terminating " character. The length of the string is the first byte at addr with the input characters following. Used in the form: " TEXT" In Interpret mode, the string will be truncated to 80 characters.
#	(d1 -- d2) Generates a double number d1, the next ASCII character which is placed in an output string. Result d2, is the quotient after division by BASE, and is maintained for further processing. Used between <# and #>. See #S.
#"	(-- value) Allows entry of 4 ASCII bytes as a literal value. Used in the form: #" ABCD would leave the hex value 41424344 on the stack.
#>	(d -- addr/d) Terminates numeric output conversion by dropping d, leaving the text address and character count suitable for TYPE.
#S	(d1 -- d2) Generates ASCII text in the text output buffer, by the use of #, until zero double number d2 results. Used between <# and #>.
'	(-- addr) Used in the form: ' nnnn Leaves the parameter field address of dictionary word nnnn. As a compiler directive, executes in a colon-definition to compile the address as a literal. If the word is not found after a search CONTEXT and CURRENT , an appropriate error message is given. Pronounced 'tick'.
(	(--) Used in the form: (cccc) Ignores a comment that will be delimited by a right parenthesis on the same line. May occur during execution or in a colon-definition. A blank after the leading parenthesis is required.
(+LOOP)	(n --) The run-time procedure compiled by +LOOP, which increments the loop index by n and tests for loop completion. See +LOOP.
(.)"	(--) The run-time procedure, compiled by ." which transmits the following in-line text to the selected output device. See ."
(;CODE)	(--) The run-time procedure, compiled by ;CODE, that rewrites the code field of the most recently defined word to point to the following machine code sequence. See ;CODE.

(CONST)	(-- value) Runtime code for CONSTANT.
(DO)	(limit/index --) The run-time procedure compiled by DO which moves the loop control parameters to the return stack. See DO .
(ECA)	(--) Compile time part for ENDCASE.
(FIND)	(addr1/addr2 -- pfa/b/ff) (ok) (addr1/addr2 -- ff) (bad) Searches the dictionary starting at the name field address addr2, matching to the text at addr1. Returns parameter field address, first byte of NFA and boolean true for a good match. If no match is found, only a boolean false is left.
(LOOP)	(--) The run-time procedure compiled by LOOP which increments the loop index and tests for loop completion. See LOOP .
(NEST)	(--) Code pointer for high level nested definition for words compiled by colon (:).
(NUMBER)	(d1/addr1 -- d2/addr2) Converts the ASCII text beginning at addr1+1 with regard to BASE . The new value is accumulated into double number d1, being left as d2. addr2 is the address of the first unconvertable digit. Used by NUMBER .
(VAR)	(-- addr) Runtime part for VARIABLE.
*	(d1/d2 -- prod) Leaves the signed product (32 bit) of two signed numbers.
*/	(n1/n2/n3 -- n4) Leaves the ratio $n4 = n1 * n2 / n3$, where all are signed numbers. Retention of an intermediate 31 bit product permits greater accuracy than would be available with the sequence: $n1\ n2\ * \ n3\ /\$
*/MOD	(n1/n2/n3 -- n4/n5) Leaves the quotient n5 and remainder n4 of the operation $n1 * n2 / n3$. A 31 bit intermediate product is used as for */.
+	(d1/d2 -- sum) Leaves the sum of d1 +d2.
+	(d/addr --) Adds n to value at the address. Pronounced 'plus-store'.
+-	(d1/d2 -- d) Applies the sign of d to d, which is left as d.

+LOOP	(d --) (run-time) (addr/d --) (compile) Used in colon-definition in the form: DO ... d +LOOP At run-time, +LOOP selectively controls branching back to the corresponding DO based on d, the loop index and the loop limit. The signed increment n is added to the index and the total compared to the limit. The branch back to DO occurs until the new index is equal to or greater than the limit (d>0), or until the new index is equal to or less than the limit (d<0). Upon exiting the loop, the parameters are discarded and execution continues ahead. At compile time, +LOOP compiles the run-time word +LOOP and the branch offset computed from HERE to the address left on the stack by DO. n is used for compile time error checking.
,	(d --) Stores d into the next available dictionary memory cell, (32 bits) advancing the dictionary pointer. (comma)
-	(d1/d2 -- diff) Leaves the difference of d1 -d2.
-FIND	(-- pfa/b/ff) (found) (-- ff) (not found) Accepts the next word (delimited by blanks) in the input stream to HERE, and searches the CONTEXT and then CURRENT vocabularies for a matching entry. If found, the dictionary entry's parameter field address, its length byte and a boolean true is left. Otherwise, only a boolean false is left.
-TRAILING	(addr/d1 -- addr/d2) Adjusts the character count n1 of a text string beginning address to suppress the output of trailing blanks. i.e. the characters at addr+d1 to addr+d2 are blanks.
.	(d --) Prints a number from a signed 32 bit two's complement value, converted according to the numeric BASE. A trailing blank follows. Pronounced 'dot'.
."	(--) Used in the form: ." cccc" Compiles an in-line string cccc (delimited by the trailing ") with an execution procedure to transmit the text to the selected output device. If executed outside a definition, ." will immediately print the text until the final ". The maximum number of characters is 255. See (.").
.4A	(value --) Displays the 4 ASCII characters represented by the input value.
.H	(value --) Displays a 4 byte Hex value.
.HB	(value --) Displays a one byte Hex value.
.HH	(value --) Displays a half byte (4 bits) Hex value.
.R	(d1/d2 --) Prints the number d1 right aligned in a field whose width is d2. No following blank is printed.

.RAM	(--) Displays information about memory usage showing main dictionary size used and free memory in user dictionary. Eg: DICTIONARY main=43650 user=48504 free=198216
.S	(--) Display the top 4 stack entries non-destructively.
/	(n1/n2 -- quot) Leaves the signed quotient of n1 /n2.
/MOD	(n1/n2 -- rem/quot) Leaves the remainder and signed quotient of n1 /n2. The remainder has the sign of the dividend.
0 1 2 3 4	(-- d) These small numbers are used so often that it is attractive to define them by name in the dictionary as constants.
0<	(d -- f) Leaves the true flag if the number is less than zero (negative), otherwise leave a false flag.
0=	(d -- f) Leaves a true flag if the number is equal to zero, otherwise, leave a false flag.
0BRANCH	(f --) The run-time procedure to conditionally branch. If f is false (zero), the following in-line parameter is added to the interpretive pointer to branch ahead or back. Compiled by IF, UNTIL and WHILE.
1+	(d1 -- d1 +1) Increments d1 by 1
2+	(d1 -- d1 +2) Increments d1 by 2
3+	(d1 -- d1 +3) Increments d1 by 3
4+	(d1 -- d1 +4) Increments d1 by 4
6+	(d1 -- d1 +6) Increments d1 by 6
8+	(d1 -- d1 +8) Increments d1 by 8
10+	(d1 -- d1 +10) Increments d1 by 10
12+	(d1 -- d1 +12) Increments d1 by 12

14+	(d1 -- d1 +14) Increments d1 by 14
16+	(d1 -- d1 +16) Increments d1 by 16
18+	(d1 -- d1 +18) Increments d1 by 18
20+	(d1 -- d1 +20) Increments d1 by 20
22+	(d1 -- d1 +22) Increments d1 by 22
24+	(d1 -- d1 +24) Increments d1 by 24
2*	(n -- n*2) 2* multiplies the number on the stack by 2. This is faster than n 2 * .
2/	(n -- n/2) 2/ divides the number on the stack by 2. This is faster than n 2 / .
4*	(n -- n*4) 4* multiplies the number on the stack by 4. This is faster than n 4 * .
4/	(n -- n/4) 4/ divides the number on the stack by 4. This is faster than n 4 / .
2DROP	(d1/d2 --) Drop the top 2 entries on the stack.
2OVER	(a/b/c/d --a/b/c/d/a/b) Drop the top 2 entries on the stack.
2SWAP	(a/b/c/d --c/d/a/b) Drop the top 2 entries on the stack.
3DROP	(d1/d2/d3 --) Drop the top 3 entries on the stack.
2DUP	(d1/d2 -- d1/d2/d1/d2) Duplicate the top 2 stack entries.
3DUP	(d1/d2/d3 -- d1/d2/d3/d1/d2/d3) Duplicate the top 3 stack entries.
7FFF	(-- 0X7FFF) Value of HEX 7FFF.
:	(--) Used in the form called colon-definition: : cccc ... ; Creates a dictionary entry defining cccc as equivalent to the following sequence of FORTH word definitions '...' until the next ';'. The compiling process is done by the text interpreter as long as STATE is non-zero. Other details are that the CONTEXT vocabulary is set to the CURRENT vocabulary and that words with the precedence bit set (P) are executed rather than being compiled.

;	(--) Terminates a colon-definition and stop further compilation. Compiles the run-time (EXIT).
<	(d1/d2 -- f) Leaves a true flag if d1 is less than d2 ; otherwise leave a false flag.
<#	(--) Setup for pictured numeric output formatting using the words: <# # #S SIGN #> the conversion is done on a double number producing text at PAD.
<<	(d1 -- d2) Logical shift of a 32 bit value by 1 bit to the left. The highest bit (bit 32) will be lost. This instruction is the same as a multiply by 2.
<<#	(d1/# of bits -- d2) Logical shift of a 32 bit value to the left by the specific number of bits. The highest bits will be lost and the low bits will be filled with zeros.
<BUILDS	(--) Used within a colon-definition: : cccc BUILDS ... DOES> ... ; Each time cccc is executed, BUILDS defines a new word with a high-level execution procedure. Executing cccc in the form: cccc nnnn uses BUILDS to create a dictionary entry for nnnn with a call to the DOES> part for nnnn. When nnnn is later executed, it has the address of its parameter area on the stack and executes the words after DOES> in cccc. BUILDS and DOES> allow run-time procedures to be written in high-level rather than in assembler code.
<CMOVE	(addr1/addr2/count --) Moves specified count of bytes starting at addr1 to memory starting at addr2. The move proceeds from high memory to low memory.
=	(d1/d2 -- f) Leaves a true flag in d1 =d2 ; otherwise leaves a false flag.
>	(d1/d2 -- f) Leaves a true flag if d1 is greater than d2 ; otherwise a false flag.
>>	(d1 -- d2) Logical shift of a 32 bit value by 1 bit to the right. The lowest bit will be lost. This instruction is the same as a divide by 2
>>#	(d1/# of bits -- d2) Logical shift of a 32 bit value to the right by the specified number of bits. The lowest bits will be lost and the highest bits will be filled with zeros.
>R	(d --) Removes a number from the computation stack and place it on the return stack. Use must be balanced with R> in the same definition.
?	(addr --) Prints the value contained at the address in free format according to the current base (equivalent to " @ . ").

?*SEARCH	(target addr/target length/key -- next addr/flag) Compares the key string anywhere within the target string and returns a flag (1 = MATCH, 0 = NO MATCH). The first byte of the key string is its length. Any ? (Hex 3F) characters in the key string will be treated as don't care positions. Next addr.contains the address of the character following the key string sequence within the target string if the flag is 1, and nothingsignificant when flag is 0.
?COMP	(--) Issues error message if not compiling
?CSP	(--) Issues error message if stack position differs from value saved in CSP.
?DUP	(d1 -- d1) (if zero) (d1 -- d1/d1) (non-zero) Reproduces d1 only if it is non-zero. This is usually used to copy a value just before IF, to eliminate the need for an ELSE to drop it.
?ERROR	(f/n --) Issues an error message number n, if the boolean flag is true.
?EXEC	(--) Issues an error message if not executing.
?MATCH	(target addr/target length3/key string addr -- flag) Compares the target string with the key string and returns a flag (1 = MATCH, 0 = NO MATCH). The first byte of the key string is its length. Any ? (Hex 3F) characters in the key string will be treated as don't care positions.
?NUMBER	(pointer to text -- tf) Returns true flag if the text is a valid number, false if not a valid number.
?PAIRS	(n1/n2 --) Issues an error message if n1 does not equal n2. The message indicates that compiled conditionals do not match.
?RANGE	(value/min/max/string addr -- value) Does a range check of a value and will display the string text if out of range.
?STACK	(--) Issues an error message if the stack is out of bounds. This definition may be installation dependent.
@	(addr -- d) Leaves the 32 bit contents of address.
ABORT	(--) Clears the stacks and enter the execution state. Control is returned to the operators terminal, printing the version message.
ABS	(d -- d) Returns the absolute value of d.

-
- AGAIN** (addr/d --) (compiling)
Used in a colon-definition in the form:
BEGIN ... AGAIN
At run-time, **AGAIN** forces execution to return to corresponding **BEGIN**. There is no effect on the stack. Execution cannot leave this loop (unless **R> DROP** is executed one level below). At compile time, **AGAIN** compiles **BRANCH** with an offset from **HERE** to addr. d is used for compile time error checking.
- ALITBUF_** (-- addr)
Used by "interpreted" to assign text buffer.
- ALLOT** (d --)
Adds the signed number to the dictionary pointer **DP**. May be used to reserve dictionary space or re-origin memory. d is the number of bytes to allot.
- ALLOTL** (d --)
Reserves dictionary space by 32 bit long words.
- ALLOTW** (d --)
Reserves dictionary space by 16 bit words.
- AND** (d1/d2 -- d3)
Leaves the bitwise logical and of d1 and d2 as d3.
- AUTONUM** (-- addr)
Control of autonumber option. Default is for **AUTONUM** to contain a 1 which will allow entry of Hex numbers with a prefix of 0X, Binary numbers with a prefix of 0B, or Octal numbers with a prefix of 0. When 0 is stored in **AUTONUM**, number entry can only be according to the current **BASE** setting.
- AUTOSTART** (--)
Loads the file **ALOAD** from disk **DR0**. This is done by the system on powerup or **REBOOT**.
- BACK** (addr --)
Calculates the backward branch offset from **HERE** to addr and compiles into the next available dictionary memory address.
- BASE** (-- addr)
A user variable containing the current numeric base used for input and output conversion.
- BEGIN** (-- addr/d) (compiling)
Occurs in a colon-definition in the form:
BEGIN ... UNTIL
BEGIN ... AGAIN
BEGIN ... WHILE ... REPEAT
At run-time, **BEGIN** marks the start of a sequence that may be repetitively executed. It serves as a return point from the corresponding **UNTIL**, **AGAIN** or **REPEAT**. When executing **UNTIL**, a return to **BEGIN** will occur if the top of the stack is false; for **AGAIN** and **REPEAT** a return to **BEGIN** always occurs. At compile time, **BEGIN** leaves its return address and n for compiler error checking.
- BIN** (--)
Sets current numerical **BASE** to binary.
-

BL	(-- value) A constant that leaves the ASCII value for 'blank'.
BLANKS	(addr/count --) Fills an area of memory beginning at addr with blanks until count is exhausted.
BLK	(-- addr) A user variable containing the address of the text being interpreted. If zero, input is being taken from the terminal input buffer.
BRANCH	(--) The run-time procedure to unconditionally branch. An in-line offset is added to the interpretive pointer IP to branch ahead or back. BRANCH is compiled by ELSE , AGAIN , REPEAT .
BTSTK	(-- value) The value is the address of the end of the computational stack.
C!	(b/addr --) Stores 8 bits at address. On word addressing computers, further specification is necessary regarding byte addressing.
C,	(b --) Stores 8 bits of b into the next available dictionary byte, advancing the dictionary pointer. This should be used with caution!
C/L	(-- value) Characters per line on the display screen. This value is 80 for the K1197 system.
C=	(b1/b2 -- f) Compare 2 byte values returning a true flag if they are equal.
C@	(addr -- b) Leaves the 8 bit contents of memory address.
CFA	(pfa -- cfa) Converts the parameter field address of a definition to its code field address.
CMOVE	(addr1/addr2/count --) Moves specified count of bytes beginning at address, addr1, to address, addr2. The contents of address addr1 is moved first proceeding toward high memory. Count is a 16 bit value.
COLD	(--) The cold start procedure to adjust the dictionary pointer to the minimum standard and restart via ABORT . May be called from the console to remove application programs and restart.

COMPILE	(--) Compile is defined Compiler word. It works in the 2. to 4. Compiler level: 1. Compiler-level: Compile-time of Forth definition containing the word COMPILE 2. level: Run-time of Forth definition, i.e. when the definition will be called. At this time the Forth definition (Parental generation) will produce another Forth definition. (Filial generation) 3. level: Compile-time of Filial definition. 4. level: Run-time of Filial definition. COMPILE compiles now in the 2. level the following Forth word in the Filial definition (3. level). In the 4. level the word following after COMPILE will be executed.
CONSTANT	(d --) A defining word used in the form: d CONSTANT cccc to create word cccc, with its parameter field containing d. When cccc is later executed, it will push the value of d to the stack.
CONTEXT	(-- addr) A user variable containing a pointer to the vocabulary within which dictionary searches will first begin.
COUNT	(addr1 -- addr2/n) Leaves the byte address addr2 and byte count n of a text string beginning at address addr1. It is presumed that the first byte at addr1 contains the text byte count and the actual text starts with the second byte. Typically, COUNT is followed by TYPE.
CR	(--) Generates a carriage return and line feed.
CREATE	(--) A defining word used in the form: CREATE cccc by such words as CODE and CONSTANT to create a dictionary header for a FORTH definition. The code field contains the address of the words parameter field. The new word is created in the CURRENT vocabulary.
CSP	(-- addr) A user variable temporarily storing the stack pointer position, for compilation error checking.
CURRENT	(-- data addr) CURRENT is a variable containing a pointer to the vocabulary to which definitions are currently being appended. CURRENT is set to point to a specific vocabulary by executing the word DEFINITIONS , which copies the vocabulary pointer in CONTEXT into CURRENT . At system startup, CURRENT is initialized to FORTH by ABORT , which is called by COLD . Note, that this means after an ABORT occurs, CURRENT will be aimed at the FORTH vocabulary.
CVT_H_STR	(pointer to text -- flag) Convert Hex literal string. The flag is set if an unconvertable character was found.

D+-	(d1/n -- d2)	
	Applies the sign of n to the double number d1, leaving it as d2.	
D.	(d --)	
	Prints a signed double number from a 32 bit values. Conversion is performed according to the current BASE. A blank follows. Pronounced 'D-dot'.	
D.R	(d/n --)	
	Prints a signed double number d right aligned in a field n characters wide.	
DABS	(d -- ud)	
	Leaves the absolute value ud of a double number.	
DECIMAL	(--)	
	Sets the numeric conversion BASE for decimal input-output.	
DEFINITIONS	(--)	
	Used in the form: cccc DEFINITIONS Sets the current vocabulary to the CONTEXT vocabulary. In the example, executing vocabulary name cccc made it the CONTEXT vocabulary and executing DEFINITIONS made both specify vocabulary cccc.	
DICTLIMIT	(-- addr)	
	The variable DICTLIMIT contains a pointer to the top of the usedictionary memory.	
DIGIT	(c/n1 -- n2/ff)	(ok)
	(c/n1 -- ff)	(bad)
	Converts the ASCII character c (using base n1) to its binary equivalent n2 ; accompanied by a true flag. If the conversion is invalid, leaves only a false flag.	
DLITERAL	(-- d)	(executing)
	(d --)	(compiling)
	If compiling, compiles a stack double number into a literal. Later execution of the definition containing the literal will push it to the stack.	
DO	(d1/d2 --)	(execute)
	(addr/d --)	(compile)
	Occurs in a colon-definition in form:	
	DO ... LOOP	
	DO ... +LOOP	
	At run-time, DO begins a sequence with repetitive execution controlled by a loop limit n1 and an index is incremented by one. Until the new index equals or exceeds the limit, execution loops back to just after DO; otherwise the loop parameters are discarded and execution continues ahead. Both d1 and d2 are determined at run-time and may be the result of other operations. Within a loop 'I' will copy the current value of the index to the stack. See I, LOOP, +LOOP, LEAVE. When compiling within the colon-definition, DO compiles (DO), leaves the following address addr and n for later error checking.	
DOES>	(--)	
	A word which defines the run-time action within a high-level defining word. DOES> alters the code field and first parameter of the new word to execute the sequence of compiled word addresses following DOES>. Used in combination with BUILDS. When the DOES> part executes it begins with the address of the first stack. This allows interpretation using this area or its contents. Typical uses include the FORTH assembler, multi-dimensional arrays and compiler generation.	

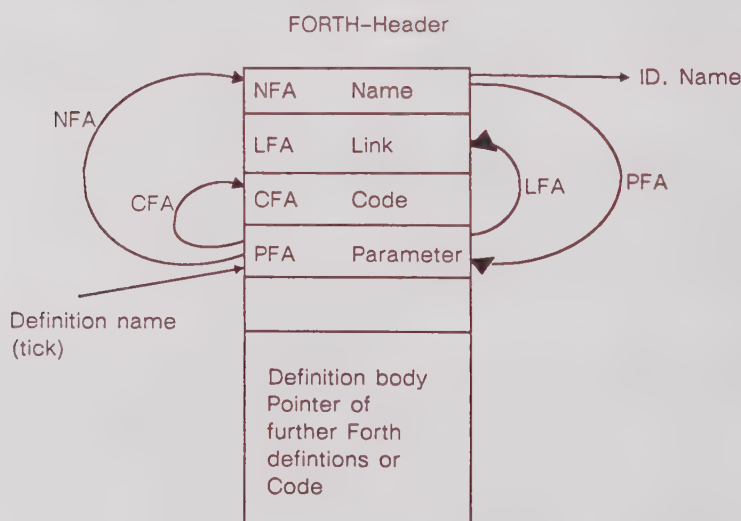
DP	(-- addr) A user variable, the dictionary pointer, which contains the address of the next free memory above the dictionary. The value may be read by HERE and altered by ALLOT .
DPL	(-- addr) A user variable containing the number of digits to the right of the decimal on double integer input. It may also be used to hold output column location of a decimal point, in user generated formatting. The default value on single number input is -1.
DR0 / DR1	(--) Selects disk drives 0 or 1
DROP	(d --) Drop the number from the stack.
DUMP	(addr/n --) Prints the contents of n memory locations beginning at addr. Both addresses and contents are shown in the current numeric base.
DUP	(d -- d/d) Duplicates the value on the stack.
ELSE	(addr1/d1 -- addr2/d2) (compiling) Occurs within a colon-definition in the form: IF ... ELSE ... ENDIF At run-time, ELSE executes after the true part following IF. ELSE forces execution to skip over the following false part and resumes execution after ENDIF. It has no stack effect. At compile time, ELSE replaces BRANCH reserving a branch offset, leaves the address addr2 and d2 for error testing. ELSE also resolves the pending forward branch from IF by calculating the offset from addr1 to HERE and storing at addr1.
EMIT	(c --) Transmit ASCII character c to the selected output device. OUT is incremented for each character output.
ENCLOSE	(addr1/c --) (addr1/c -- addr1/d1/d2/d3) The text scanning primitive used by WORD . From the text addr1 and an ASCII delimiting character c, is determined the byte offset to the first non-delimiter character d1, the offset to the first delimiter after the text d2, and the offset to the first character not included. This procedure will not process past an ASCII 'null', treating it as an unconditional delimiter, except when the delimiter is ')' or ''.
ENDIF	(addr/d --) Alias for THEN . It is used to end the IF ELSE construct.
ERASE	(addr/d --) Sets a region of memory to zero from addr over d bytes.
ERROR	(n --) Executes error notification and restart of system. Error number n is displayed.

EXECUTE	(addr --) Executes the definition whose code field address is on the stack.
EXIT	(--) EXIT is the run-time word compiled at the end of a colon-definition which returns execution to the calling procedure.
EXPECT	(addr/count --) Transfers characters from the terminal to address, until a "return" or the count of characters have been received. One or more nulls are added at the end of the text. WARNING: EXPECT clears the buffer with spaces before it starts collecting characters.
FENCE	(-- addr) A user variable containing an address below which FORGETting is trapped.
FFFFFFFFE	(-- 0xFFFFFFFF) 32 bit hex value used to mask even values.
FILL	(addr/quant/b --) Fill memory at the address with the specified quantity of bytes b.
FILLW	(addr/quant/w --) Fill memory at the address with the specified quantity of 16 bit words with a value of w.
FLD	(-- addr) A user variable for control of number output field width. Presently unused.
FORGET	(--) Executed in the form: FORGET cccc Deletes definition named cccc from the dictionary with all entries physically following it. In fig-FORTH, an error message will occur if the CURRENT and CONTEXT vocabularies are not currently the same.
FORTH	(--) The name of the primary vocabulary. Execution makes FORTH the CONTEXT vocabulary. Until additional user vocabularies are defined, new user definitions become a part of FORTH. FORTH is immediate, so it will execute during the creation of a colon-definition, to select this vocabulary at compile time.
GOFORTH	(--) GOFORTH does a warm start of the system. Any program execution will be halted and the FORTH interpreter will be active.
HERE	(-- addr) Leaves the address of the next available dictionary location.
HEX	(--) Set the numeric conversion base to sixteen (HEXadecimal).
HLD	(-- addr) A user variable that holds the address of the latest character of text during numeric output conversion.

HLITBUF	(-- addr) Used by interpreted X" to assign text buffer.
HOLD	(c --) Used between <# and #> to insert an ASCII character into a pictured numeric output string. eg. 2E HOLD will place a decimal point.
I	(-- d) Used within a DO-LOOP to copy the loop index to the stack. Other use is implementation dependent. See R.
ID.	(addr --) Prints a definition's name from its name field address. (addr)
IF	(f --) (run-time) (-- addr/d) (compile) Occurs in a colon-definition in the form: IF (tp) ... ENDIF IF (tp) ... ELSE (fp) ... ENDIF At run-time, IF selects execution based on a boolean flag. If f is true (non-zero), execution continues ahead through the true part. If f is false (zero), execution skips until just after ELSE to execute the false part. After either part, execution resumes after ENDIF. ELSE and its false part are optional. If missing, false execution skips to just after ENDIF. At compile time, IF compiles OBRANCH and reserves spac for an offset at addr. addr and d are used later for resolution of the offset and error testing.
IMMEDIATE	(--) Marks the most recently made definition so that when encountered at compile time, it will be executed rather than being compiled. i.e. the precedence bit in its header is set. This method allows definitions to handle unusual compiling situations, rather than build them into the fundamental compiler. The user may force compilation of an immediate definition preceeding it with (COMPILE).
IN	(-- addr) A user variable containing the byte offset within the current input text buffer (terminal or disc) from which the next text will be accepted. WORD uses and moves the value of IN.
INTERPRET	(--) The outer text interpreter which sequentially executes or compiles text from the input stream (terminal or disc) depending on STATE. If the word name cannot be found after a search of CONTEXT and then CURRENT, it is converted to a number according to the current base. That also failing, an error message echoing the name with a '?' will be given. Text input will be taken according to the convention for WORD.
J	(-- d) Leaves the outer loop index in a double-nested DO LOOP
KBYTE	(n -- nkbyte) Leaves n *1024 on the stack.
KEY	(-- c) Leaves the ASCII value of the next terminal key struck.

L!	(value/addr --) Same function as ! (store). Stores 32 bit value at address.
L@	(addr -- value) Same function as @ (fetch). Fetches 32 bit value from address.
LATEST	(-- addr) Leaves the name field address of the top-most word in the CURRENT vocabulary.
LEAVE	(--) Forces termination of a DO-LOOP at the next opportunity by setting the loop limit equal to the current value of the index. The index itself remains unchanged, and execution proceeds normally until LOOP or +LOOP is encountered.
LFA	(pfa -- lfa) Converts the parameter field address of a dictionary definition to its link field address.
LIT	(-- d) Within a colon-definition, LIT is automatically compiled before each 32 bit literal number encountered in input text. Later execution of LIT causes the contents of the next dictionary address to be pushed to the stack.
LITERAL	(d --) (compiling) If compiling, then compile the stack value d as 32 bit literal. This definition is immediate so that it will execute during a colon-definition. The intended use is: : xxx (calculate) LITERAL ; Compilation is suspended for the compile time calculation of a value. Compilation is resumed and LITERAL compiles this value.
LOOP	(addr/d --) (compiling) Occurs in a colon-definition in the form: DO ... LOOP At run-time, LOOP selectively controls branching back to the corresponding DO based on the loop index and limit. The loop index is incremented by one and compared to the limit. The branch back to DO occurs until the index equals or exceeds the limit. At that time, the parameters are discarded and execution continues ahead. At compile time, LOOP compiles (LOOP) and uses addr to calculate an offset to DO. d is used for error checking.
M*	(d1/d2 -- d) A math operation which leaves the signed product of two signed numbers.
M/	(d/n2 -- n3/n3) A math operator which leaves the signed remainder n2 and signed quotient n3, from a double number dividend and divisor n1. The remainder takes its sign from the dividend.
M/MOD	(ud1/u2 -- u3/ud4) An unsigned math operation which leaves a double quotient ud 4 and remainder u3, from a double dividend ud1 and single divisor u2.
MAX	(d1/d2 -- max) Leaves the greater of two numbers.

- MIN** (d1/d2 -- min)
Leaves the smaller of two numbers.
- MOD** (n1/n2 -- mod)
Leaves the remainder of n1 /n2, with the same sign as n1 .
- MOVE** (addr1/addr2/n --)
Moves the contents of n memory cells (16 bit contents) beginning at addr1 into n cells beginning at addr2 .The contents of addr1 is moved first.
- N->S** (addr of last char/number -- addr of first char)
N->S does a number to ASCII string conversion. The conversion will proceed from the last character to the first character, returning the address of the first character.
- NEGATE** (d1 -- d2)
Leaves the two's complement of a number.
- NFA** (pfa -- nfa)
Converts the parameter field address of a definition to its name field.



- NO** (-- 0)
Constant with a value of 0, used when a false flag is desired.
- NULL_TERM** (addr/"string" --)
Null terminates the string and stores it at the address.
- NUMBER** (addr -- d)
Converts a character string (terminated by an ASCII blank!) at addr with a preceding count, to a signed number, using the current numeric base. If numeric conversion is not possible, an error message will be given.
- OCTAL** (--)
OCTAL sets the number BASE to octal.

OFF	(-- 0) OFF is a constant with a value of 0.
ON	(-- 1) ON is a constant with a value of 1.
OR	(d1/d2 -- or) Leaves the bit-wise logical or of two 32 bit values.
OUT	(-- addr) A user variable that contains a value incremented by EMIT. The user may alter and examine OUT to control display formatting.
OVER	(d1/d2 -- d1/d2/d1) Copies the second stack value, placing it as the new top.
PAD	(-- addr) Leaves the address of the text output buffer, which is fixed offset above HERE.
PFA	(nfa -- pfa) Converts the name field address of a compiled definition to its parameter field address.
PICK	(n -- copy of nth stack entry) Leaves a copy of the nth stack entry. N starts at 0, thus 0 PICK is the same as DUP.
PREFDOT	(-- addr) PREFDOT is a variable that controls the prefix for the . (dot) number display. When PREFDOT is set to 0, no prefix is done to number display. When PREFDOT is set to 1 and BASE is set to Hex, the . will add a prefix of 0X. When PREFDOT is set to OCTAL, the . will add prefix of 0C.
QUERY	(--) Inputs 80 characters of text (or until a 'return') from the operators terminal. Text is positioned at the address contained in TIB with IN set to zero.
QUIT	(--) Clears the return stack, stops compilation, and returns control to the operators terminal. No message is given.
R	(-- d) Copies the top of the return stack to the computation stack.
R#	(-- addr) A user variable which may contain the location of an editing cursor, or other file related function.
R>	(-- d) Removes the top value from the return stack and leaves it on the computation stack. See >R and R.
R0	(-- addr) A user variable containing the initial location fo the return stack. Pronounced R-zero. See RP!

REPEAT	(addr/d --) (compiling) Used within a colon-definition in the form: BEGIN ... WHILE ... REPEAT At run-time, REPEAT forces an unconditional branch back to <code>jus</code> after the corresponding <code>BEGIN</code> . At compile time, REPEAT compiles <code>BRANCH</code> and the offset from <code>HERE</code> to <code>addr</code> . <code>d</code> is used for error checking.
ROT	(d1/d2/d3 -- d2/d3/d1) Rotates the top three values on the stack, bringing the third to the top.
RP!	(--) Initialize the return stack pointer from user variable <code>R0</code> .
RP@	(--) Return stack address.
S->N	(startaddr -- 0 value / 1) <code>N->S</code> does ASCII string conversion to a number. Example: " 123" -> 123 1 " ABC" -> 0
S.	(d -- d) Displays the numerical value of the top stack entry without removing it.
S0	(-- addr) A user variable that contains the initial value for the stack pointer. Pronounced S-zero. See <code>SP!</code>
SC@	(addr -- signed b) Leaves the 8 bit contents of memory address with sign extension. (Fetch with sign extension)
SIGN	(n/d -- d) Stores an ASCII '-' sign just before a converted numeric output string in the text output buffer when <code>n</code> is negative. <code>n</code> is discarded, but double number <code>d</code> is maintained. Must be used between <code><#</code> and <code>#></code> .
SMUDGE	(--) Used during word definition to toggle the 'smudge bit' in a definition's name field. This prevents an uncompleted definition from being found during dictionary searches, until compiling is completed without error.
SP!	(resets) Initialize the stack pointer from <code>S0</code> .
SP@	(-- addr) Return the address of the stack position to the top of the stack, as it was before <code>SP@</code> was executed. (eg. 1 2 SP@ @ . . . would type 2 2 1).
SPACE	(--) Transmits an ASCII blank to the screen.
SPACES	(n --) Transmits <code>n</code> ASCII blanks to the screen.

STACK	(--)	Non-destructive display of all stack contents with top stack entry on the right.
STATE	(-- addr)	A user variable containing the compilation state. A non-zero indicates compilation. The value itself may be implementation dependent.
SW@	(addr -- signed n)	Leaves the 16 bit contents of memory address with sign extension. (Word Fetch with sign extension)
SWAP	(d1/d2 -- d2/d1)	Exchanges the top two values on the stack.
SWAPB	(b1/b2 --b2/b1)	Swaps the lower and upper two bytes on a 16-bit word.
THEN	(addr/d --) (compile)	Occurs in a colon-definition in form: IF ... THEN IF ... ELSE ... THEN At run-time, THEN serves only as the destination of a forward branch from IF or ELSE. It marks the conclusion of the conditional structure. See also IF and ELSE. At compile time, THEN computes the forward branch offset from addr to HERE and stores it addr. d is used for error tests.
TIB	(-- addr)	A user variable containing the address of the terminal input buffer.
TOGGLE	(addr/b --)	Complements the contents of the address by the bit pattern b.
TOGGLEW	(addr/n --)	Complements the contents of the address by the 16 bit pattern n.
TYPE	(addr/count --)	Transmits count characters from addr to the screen.
TYPH	(addr/count --)	Transmits count characters from addr to the screen and displays them in hex format.
U*	(u1/u2 -- ud)	Leaves the unsigned numeric product of two unsigned numbers.
U/	(ud/u1 -- u2/u3)	Leaves the unsigned remainder u2 and unsigned quotient u3 from the unsigned double dividend ud and unsigned divisor u1 .
UNTIL	(f --) (run-time) (addr/d --) (compile)	Occurs within a colon-definition in the form: BEGIN ... UNTIL At run-time, UNTIL controls the conditional branch back to the corresponding BEGIN. If f is false, execution returns to just after BEGIN; if true, execution continues ahead. At compile time, UNTIL compiles (OBRANCH) and an offset from HERE to addr. d is used for error tests.

VARIABLE	(d --) A defining word used in the form: d VARIABLE cccc When VARIABLE is executed, it creates the definition cccc with its parameter field initialized to d. When cccc is later executed, the address of its parameter field (containing n) is left on the stack, so that a fetch or store may access this location.
VLIST	(--) Lists the names of the definitions in the context vocabulary.
VOC-LINK	(-- addr) A user variable containing the address of a field in the definition of the most recently created vocabulary. All vocabulary names are linked by these fields to allow control for FORGETTING through multiple vocabularies.
VOCABULARY	(--) A defining word used in the form: VOCABULARY cccc to create a vocabulary definition cccc. Subsequent use of cccc will make it the CONTEXT vocabulary which is searched first by INTERPRET. The sequence " cccc DEFINITIONS" will also make cccc the CURRENT vocabulary into which new definitions are placed. In fig-FORTH, cccc will be so chained as to include all definitions of the vocabulary in which cccc is itself defined. All vocabularies ultimately chain to FORTH. By convention, vocabulary names are to be declared IMMEDIATE. See VOC-LINK.
VOCIN	(--) Sets the initial vocabulary pointer to FORTH.
W!	(n/addr --) Stores 16 bits of n at address. Pronounced 'word store'.
W,	(n --) Stores n into the next available dictionary memory cell (16 bits) advancing the dictionary pointer. Pronounced 'word comma'.
W<	(n1/n2 -- f) Leaves a true flag if n1 is less than n2 ; otherwise, leaves a false flag. (16 bit values)
W=	(n1/n2 -- f) Leaves a true flag if n1 = n2 ; otherwise, leaves a false flag (16 bit values)
W>	(n1/n2 -- f) Leaves a true flag if n1 is greater than n2 ; otherwise, leaves a false flag. (16 bit values)
W@	(addr -- n) Leaves the 16 bit contents of address. (word fetch)
WCMOVE	(src/dst/length --) Source string is in word format and is moved to destination address with high bytes deleted.

WHILE	(f --) (run-time) (ad1/d1 -- ad1/d1/ad2/d2) (compile) Occurs in a colon-definition in the form: BEGIN ... WHILE (tp) ... REPEAT At run-time, WHILE selects conditional execution based on boolean flag f. If f is true (non-zero), WHILE continues execution of the true part through REPEAT, which then branches back to BEGIN. If f is false (zero), execution skips to just after REPEAT, exiting the structure. At compile time, WHILE replaces (0BRANCH) and leaves ad2 of the reserved offset. The stack values will be resolved by REPEAT.
WIDTH	(-- addr) In fig-FORTH, a user variable containing the maximum number of letters saved in the compilation of a definition's name. It must be 1 through 31, with a default value of 31. The name character count and its natural characters are saved, up to the values in WIDTH.
WORD	(c --) parse one word from input to HERE
X"	(-- addr) Hex string input. Used in the form of X" Hexstring". The length of the string is in the first byte at addr with the byte string following.
XOR	(d1/d2 -- xor) Leaves the bit wise logical exclusive-or of two values.
YES	(-- 1) Constant with a value of 1 used when true flag is desired.
[	(--) Used in colon-definitions in the form: : xxx [words] more ; Suspends compilation. The words after [are executed, not compiled. This allows calculation or compilation exceptions before resuming compilation with]. See LITERAL,].
[COMPILE]	(--) Used in a colon-definition in the form: : xxx [COMPILE] FORTH ; [COMPILE] will force the compilation of an immediate definition, that would otherwise execute during compilation. The above example will select the FORTH vocabulary when xxx executes, rather than at compile time.
]	(--) Resumes compilation, to the completion of a colon-definition. See [.

4. K1197 FORTH ENHANCEMENTS

4.1. INTRODUCTION

This document describes the enhancements to the FORTH language provided by Siemens to make data communications testing possible. The enhancements provide the necessary mechanism to enable test programs to interface to the run-time environment (referred to as RTE from here on) modules.

4.2. SYNOPSIS

The interface between the FORTH program (including enhancements) and the RTE is accomplished by front ends between these two sections. This is shown in Fig. 4-1 on page 4-1

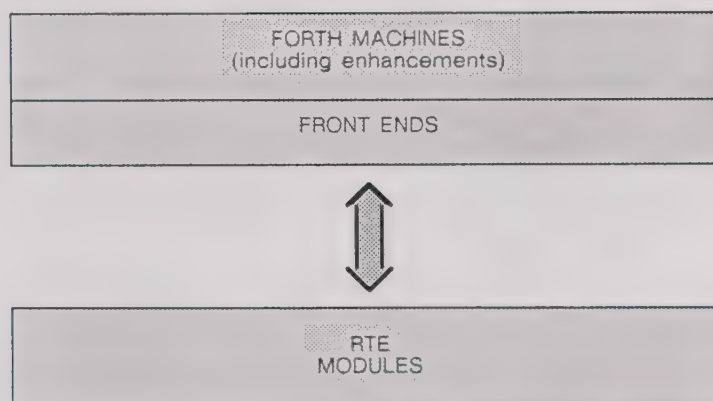


Figure 4-1 FORTH/RTE INTERFACE

FORTH machines are:

monitor, analysis or emulation program for
example: X.21, X.25, X.75, SNA, SDLC, BSC 3270, ISDN, CCS#7.

4.2.1. COMMAND SUMMARY

The following summarizes the various command categories required to extend FORTH to provide data communications testing capability.

- General Definitions
- Miscellaneous
- Case Construct
- MAKE and DOER Construct
- Conditional Compile Operations
- Interface Lead Control and Checking
- K1197 Interface Leads
- Beeper Control
- Timer
- Time
- Disk Interface
- Display Manager
- Display
- M Display Routines
- Cursor Positioning
- Function Key Manager
- Keyboard Input
- Error Recovery and Error Protection
- Capture Buffer Control
- Disk Playback Control
- Printer Port
- Remote Modem Port
- Configuration
- Data Oriented
- State Machine Related
- Code Conversion
- CRC Calculation

4.3. GENERAL DEFINITIONS

PORT DEFINITIONS

LOCAL	(-- value) Color CRT (memory mapped) and K1197 keyboard
PRINTER	(-- value) Serial async. port (to DTE configuration)
REMOTE	(-- value) Serial async. port (to DCE configuration)
PORT1 PORT1_A PORT1_B	(-- value) (-- value) (-- value) PORT1 is the port identifier for the top interface connectors. This is the only test port that is available on a single port machine. PORT1_A and PORT1_B are the transmitter and receiver identifiers for PORT1. PORT1_A PORT1_B OR is equal to PORT1.
PORT2 PORT2_A PORT2_B	(-- value) (-- value) (-- value) PORT2 is the port identifier for the top interface connectors. This port is not available on a single port machine. PORT2_A and PORT2_B are the transmitter and receiver identifiers for PORT2. PORT2_A PORT2_B OR is equal to PORT2.
PORT3 PORT3_A PORT3_B PORT4 PORT4_A PORT4_B	(-- value) (-- value) (-- value) (-- value) (-- value) (-- value) Returns the ID for port 3 (combined port 3a and port 3b), port 3a, port 3b, port 4 (combined port 4a and port 4b), port 4a, port 4b, port 5 (combined port 5a and port 5b), port 5a, port 5b, port 6 (combined port 6a and port 6b), port 6a and port 6b
MYPORT	(-- id_of_port_a_and_port_b_for_that_FORTH_machine)
MYPORTA	(-- id_of_port_a_for_that_FORTH_machine)
MYPORTB	(-- id_of_port_b_for_that_FORTH_machine) Return the id of the port that logically ties to the FORTH machine. Both channels of PORT 1 are logically tied to the MAIN and FORTH machine 1, PORT 2 is assigned to FORTH machine 2, PORT 3 is assigned to FORTH machine 3 and so on.

NUMBERING CONVENTIONS

To avoid potential conflicts arising from the user failing to enter all necessary arguments, a numbering convention is adopted that places arguments in specific ranges, and all commands in a

specific range. In this way the compiler can do a range check to ensure that no required arguments have been omitted. In cases where fields in commands are assigned numeric values, these fields will be given values which are powers of two. This will facilitate multiple sources or destinations by use of the logical 'OR' or '+' operation.

FORTH DATA CONVENTIONS

FORTH stack manipulations involve operating on 32-bit data units. This is the standard here. Whenever smaller units are used, they appear on the stack right justified.

FORTH NAMING CONVENTIONS

Names may contain up to 31 characters. All characters except 'space' and 'carriage return' are permitted.

Note: These naming conventions do not apply to disk filenames.

DATA FORMATS

1. Data sizes

byte – 8 bit byte or character.

FORTH words beginning with 'C' normally deal with bytes eg. C@,C!

word – 16 bits.

FORTH words beginning with 'W' normally deal with 16-bit words eg. W@

long word – 32 bits.

FORTH words beginning with 'L' or 'D' normally deal with 32 bits. This is the default.

2. Literals

Numeric literals, maximum of 32 bits.

These are decimal numbers and must not begin with 0.

Prefixes are 0X for HEX, 0 for octal and 0B for binary.

Short immediate ASCII literals eg. # "text",
where text may be up to 4 characters in length.

Long "pointer" literal eg. " long text",
where a maximum of 255 characters in compile mode are passed by a pointer on a stack.

Long HEX "pointer" literal eg. X "033F",
where HEX character pairs are entered. An odd number of characters will result in the last half digit in the high nibble.

3. Strings

type 1 – may be accessed by address and length eg. ADR 12 TYPE

type 2 – may be accessed by address, pointing to the length byte followed by the string eg. ADR COUNT TYPE

type 3 – short strings of max. 32 bits, passed from # " or to .4A on stack.

DATA TYPES

Constants	Fixed values which are placed on the stack directly.
Variables	Values whose address are placed on the stack. They may be used for vectors, matrixes, conversion conversion tables, arrays, data fields, buffers, etc... (similar to C pointers).
Words	FORTH words may, in general, use any format for stack activity.
FILE NAMES	Up to 14 alpha-numeric characters.

4.4. MISCELLANEOUS COMMANDS

MENU	(--) Returns the system to the main menu page. All open receive channels are closed. MENU is compiled as the first section of the user dictionary (disk file ALOAD).
EXEC	(filename --) Reads and compiles the specified program. Where filename is the name of the program to be executed. NOTE: Filename leaves the address of the filename on the stack, therefore another filename entry mechanism could be utilized provided the address is left on the stack.
EXEC_ERR	(-- tf) Variable which contains the result of the last EXEC or EXECB operation. Where the fetch operation returns: 0 = success 1 = failure
SAVEB	(filename --) Saves the present user words (everything above TOP-MAIN-DICT) in the specified file. The file may be read with EXECB. Where filename is the name of the file under which the binary dictionary is stored. If the filename already exists in the directory, it will be unconditionally removed. If the filename ALOAD is assigned a binary file on a systemdisk, then this file will automatically be loaded after the operating system (and automatically started if the entry point is also called ALOAD).
EXECB	(filename --) Loads the file saved with the SAVEB command into the dictionary. NOTE: The previous user dictionary will be overwritten. The command also verifies that the operating system presently in the K1197 is compatible with the one used to save the dictionary.
#_LOOPS	(-- addr) Variable used by EXECB and SAVEB.

RUN_PROGRAM (filename --)

Runs the program whose name is on the stack. The command execution attempts to find the name in the dictionary. If successful, the program is automatically started. If not successful, the command looks for a filename on drive 0 to match the entered name. If it is found, the binary file is loaded into the dictionary, and providing the entry point matches the name, the program is automatically started. If the entry point does not match, control is returned to the FORTH interpreter so that the program can be started manually. If the filename was not found on drive 0, the above procedure is repeated for drive 1. For units with a Winchester Disk Drive, the above procedure is repeated for all logical drives currently supported.

Where filename is the filename and/or program entry point.

FILE_NAME (-- addr)

FILE_NAME returns the address of the 16 byte array used to hold the disk file name by RUN_PROGRAM .

EDIT (--)

Calls up the EDITOR FUNCTION.

NOTE: This command must be used with extreme caution. The EDITOR FUNCTION takes over management of the keyboard and display. When the editing session is terminated, the calling program must re-establish keyboard and display management. In other words, EDIT should not be entered on the command line.

EDITF (filename --)

Calls up the EDITOR FUNCTION, and automatically reads the specified file into the edit buffer.

NOTE: This command must be used with extreme caution. The EDITOR FUNCTION takes over management of the keyboard and display. When the editing session is terminated, the calling program must reestablish keyboard and display management.

Where filename is the name of the file and is entered in the form " FILENAME".

EDITM (--)

Calls up the EDITOR FUNCTION using the content of the EDITOR-BUFFER. No Disk Operations are performed. This command is useful if you left the EDITOR mistakenly without saving the results.

EDITBUF (-- addr)

Returns the starting address of the EDITOR buffer. The buffer has a size of 64 kilobytes. This buffer is also used by EXEC, COPY and COMPARE and for the FAST-MODE.

REBOOT (--)

Causes complete rebooting of the system disk. The system will startup as if the power had just been turned on (but without doing the SELF-TEST).

AUTO_DISK_RPT (-- addr)

AUTO_DISK_RPT is the flag which tells the system to report status after any disk operation. The default is NO.

YES AUTO_DISK_RPT !

causes the system to perform a REPORT after every disk operation.

NOTE: This variable is common among all FORTH machines.

AUTO_REBOOT (-- addr)

AUTO_REBOOT is a variable that is used by the system to decide to reboot after a crash.

NO AUTO_REBOOT ! causes the system not to reboot after it crashed.

YES AUTO_REBOOT ! causes the system to reboot after it crashes. This is the default setting.

NOTE: This variable is common among all FORTH machines.

DSMBUF (-- addr)

Returns the starting address of the data recording buffer. This buffer is 24 kbytes long. This buffer is also used by DIR, DIRM and PLAYBACK.

ASTART_ (-- addr)

When the ASTART_ variable is 1, the system will load the file ALOAD and execute the word ALOAD automatically after powerup.

RUN_RTE (--)

Runs the other functions in the system and returns when it is done.

FIND_WORD (text addr -- pfa/b/ff) (OK)
(text addr -- ff) (bad)

Searches the dictionary for a match to the text at the input address. Returns parameter field address, first byte of NFA and true flag (1) for a good match. If no match is found, only a false flag (0) is returned.

EXECUTE_WORD (string addr --)

Searches the dictionary for the word specified by the string and executes it if found. If the word is not found, nothing will be done. The string must have its length as the first byte at the address. It can be used in the form: " WORD" EXECUTE_WORD.

RUN_WORD (string addr --)

Searches the dictionary for the word specified by the string and executes it if found. Both the computational stack and return stack will be cleared. If the word is not found, nothing will be done. The string must have its length as the first byte at the address. Can be used in the form: " WORD" RUN_WORD.

MBOX-0 to (-- addr)

MBOX-9 Variable used by application software to pass parameters between them.

?VERSION (-- REVNUMBER / VERSIONNUMBER / VERSIONNUMBER / ADDRESS)

Returns the Operating System identification:

1. Revision number
2. Version number (post)
3. Version number (pre)
4. Address of one of identification strings

INTERFACE (-- interface-type)
 returns local Interface type (not available on MAIN)

0	#V/X
1	#BRI
2	#PRI

INTERFACE1
 INTERFACE2 (-- interface-type)
 returns local Interface type of the interface

CONNECTOR (-- connector type)
 returns the local connector type(not available on MAIN)

0	V/X
1	S0
2	UP0
3	S2M
4	T1

CONNECTOR1
 CONNECTOR2 (-- connector-type)
 returns local connector type of the interface

IP0 (-- addr)
 IP1 (-- addr)
 IP2 (-- addr)
 IP3 (-- addr)
 IP4 (-- addr)
 IP5 (-- addr)
 IP6 (-- addr)
 IP7 (-- addr)
 IP8 (-- addr)

These variables are created so that parameters can be passed between FORTH machines. These words return the addr of the variable. IP0, IP1, ... , IP8 give an address on main, CPU1,...,CPU8 respectively. The error message "Nonexistent Processor" is printed if its corresponding CPU does not exist and the addr of the local IP variable is returned.


```

ENDRAM      ( -- addr )
ENDRAM_1    ( -- addr )
ENDRAM_2    ( -- addr )
ENDRAM_3    ( -- addr )
ENDRAM_4    ( -- addr )
ENDRAM_5    ( -- addr )
ENDRAM_6    ( -- addr )
ENDRAM_7    ( -- addr )
ENDRAM_8    ( -- addr )
ENDRAM_M    ( -- addr )

```

These words return the address of the first byte past the end of the local CPU, main CPU, CPU1, CPU2, CPU3, CPU4, CPU5, CPU6, CPU7 and CPU8 ram respectively. The error message "Nonexistent Processor" is printed out if its corresponding cpu does not exist and the address returned is that of the first byte past the end of memory of the local cpu.

The error message "nonexistent processor" is printed if its corresponding cpu does not exist and the address of the first byte past the end of the memory of the local cpu is returned.

```

RTEEND      ( -- addr )
RTEEND_1    ( -- addr )
RTEEND_2    ( -- addr )
RTEEND_3    ( -- addr )
RTEEND_4    ( -- addr )
RTEEND_5    ( -- addr )
RTEEND_6    ( -- addr )
RTEEND_7    ( -- addr )
RTEEND_8    ( -- addr )
RTEEND_M    ( -- addr )

```

These words return the address of the first byte past the end of the local CPU, main CPU, CPU1, CPU2, CPU3, CPU4, CPU5, CPU6, CPU7 and CPU8's system code respectively. The error message "Nonexistent Processor" is printed out if its corresponding cpu does not exist and the address of the first byte past the end of the local cpu's system code is returned.

4.5. CASE CONSTRUCT

The case construct has been added to provide a mechanism for selecting one of a set of possible cases. It permits straightforward implementation of state machines and generally allows operations found in languages like C. There are 6 words associated with the case construct:

- i DOCASE
- ii CASE
- iii ORCASE
- iv {
- v }
- vi ENDCASE

```

DOCASE      ( -- )
             Starts the case construct. The case value must be on the stack prior to usage.

```

```

CASE        ( -- )
             Initiates the comparison of the value on the stack with the result of the sequence
             between the CASE and { words.

```

ORCASE (--)
An extension of the case to permit multiple selections to have the same execution sequence.

{ (--)
Starts the execution sequence if the case comparison resulted in a TRUE condition.

} (--)
Terminates the execution sequence.

ENDCASE (--)
Terminates the case construct.

Example:

```
: TEST_CASE ( VALUE -- )  
  
  DOCASE  
    CASE 1 { ." VALUE=1" }  
    CASE 2 { ." VALUE=2" }  
    CASE 3  
    ORCASE 4 { ." VALUE=3 OR 4" }  
    CASE DUP { ." OTHER VALUE" }  
  ENDCASE  
;
```

The CASE construct must always be used inside a word definition. The TEST_CASE word is given as an example of usage.

The value to be compared (equivalent to the switch in C) must be on the stack prior to execution of the DOCASE word. This value is then compared to the values given between the CASE and {. If the two values are equal, the words between the { and the } will be executed with execution then continuing following the ENDCASE word. ORCASE can be used to give multiple choices with the same execution sequence. A default condition can be created by using DUP instead of a value in the last CASE line.

4.6. MAKE AND DOER CONSTRUCT

This construct allows vectorable FORTH words to be defined.

DOER	(--) Defines a word whose behaviour is vectorable by using MAKE
MAKE	(--) Vectors the doer-name word to the FORTH code. The doer-name word must have been created using DOER . Used interpretively: MAKE doer-name forth-code ; or inside a definition: : def ... MAKE doer-name forth-code ;AND ... ;
;AND	(--) Allows continuation of the "making" definition when MAKE is used inside of a colon definition.
NOTHING	(--) A do nothing word used by DOER .

Note: The implementation of **MAKE** and **DOER** words are based on the description of **MAKE** and **DOER** in the book – *THINKING FORTH* – by Leo Brodie.

Warning: The using of the **MAKE** command without corresponding ; or ;AND will crash the system.

Example:

	DOER show_number	
	MAKE show_number . ;	Display on screen
or	MAKE show_number P. ;	Display on printer
or	MAKE show_number DUP . P. ;	Display on screen or printer

4.7. CONDITIONAL COMPILE OPERATIONS

The conditional compile operations allow FORTH words to be compiled conditionally. This has the advantage of keeping the source for different versions in one file.

#IFDEF	(--) #IFDEF checks whether the identifier (FORTH word) is currently defined in the dictionary. If so, the code following this #IFDEF up to the corresponding #ELSE or #ENDIF will be compiled. Otherwise, the code is skipped. #IFDEF can be nested.
#ELSE	(--) #ELSE is the false part of #IFDEF. The code following this #ELSE up to the corresponding #ENDIF will be compiled.
#ENDIF	(--) #ENDIF marks the end of the #IFDEF conditional compile.

#IFNOTDEF (--)

#IFNOTDEF checks whether the identifier (FORTH word) is currently undefined in the dictionary. If so, the code following this #IFNOTDEF up the corresponding #ELSE or #ENDIF will be compiled. Otherwise, the code is skipped. #IFNOTDEF can be nested.

#IF (-- state)

#IF allows conditional compilation based upon the returned value of an enclosed expression. The expression must leave a single value on the stack and enclosed by either #IS_TRUE or #IS_FALSE. #IF may be used within a colon definition and is to be used with #ELSE and #ENDIF.

#IS_TRUE (state/flag --)

#IS_TRUE restores the compiling or executing state of the FORTH machine. If the flag is non-zero (true), then the code after #IF_TRUE is executed. If the flag is zero (false), then the code following #IS_TRUE is skipped until #ELSE or #ENDIF is encountered. State, is the state returned by #IF and flag is the value returned by the enclosed expression after #IF.

#IS_FALSE (state/flag --)

#IS_FALSE is the complement of #IS_TRUE and is provided for readability of source code. It is functionally equivalent to 0= #IS_TRUE. State is the state returned by #IF and flag is the value returned by the enclosed expression after #IF.

Example

```
#IF A @ B @ = #IS_TRUE  
." A IS EQUAL B" CR  
#ENDIF
```

(If the variables A and B are equal the jmessage will be displayed. You can use the conditions also with execution commands.)

4.8. INTERFACE LEAD CONTROL AND CHECKING

LIST OF INTERFACE LEAD CONSTANTS

The following list contains the interface leads. For **TURN_ON** and **TURN_OFF**, the location of the driver is indicated by an X in the appropriate column. For the **CHECK** operation, it is possible to check both the received status as well as the driven status by specifying the relevant port number.

TO_DCE	TO_DTE	Interface Lead	
X		IRS	(-- d) - request to send (RTS)
	X	ICS	(-- d) - clear to send (CTS)
	X	IDM	(-- d) - data set ready (DSR)
X		ITR	(-- d) - data terminal ready (DTR)
	X	IRR	(-- d) - carrier detect (DCD)
	X	ISQ	(-- d) - signal quality (SQ)
X		ISR	(-- d) - data signal rate select to DCE
X		ISS	(-- d) - select DCE standby facility
	X	IIC	(-- d) - ring indicate (RI)
X		ILL	(-- d) - local loopback (LL)
X		IIS	(-- d) - terminal in service
X		IC	(-- d) - control (V.11) (C)
	X	II	(-- d) - indication (V.11) (I)
	X	IR	(-- d) - receive (V.11) (R)
X		IT	(-- d) - transmit (V.11) (T)

WORDS

TURN_ON (port/lead --)
Turns on the interface control lead. Where port may be **PORT1** or **PORT2** and where INTERFACE LEAD NAME may be chosen from the above list.

TURN_OFF (port/lead --)
Turns off the interface control lead.
Where port may be **PORT1** or **PORT2**.

Warning: Turning on/off of Receive (R) or Transmit(T) control leads is not possible.

CHECK (port/lead --)
Tests the interface lead. Returns 1=ON, 0=OFF. If an invalid lead name has been entered, -1 will be returned.
Where port may be **PORT1** or **PORT2**.

LEAD_DISPLAY_OFF (port --)
Disables display of interface lead status.

LEAD_DISPLAY_ON (port/addr --)
Enables display of interface lead status. Two data streams may be selected and the display will be taken care of automatically on the screen and specified by the user. The display format is determined by the interface module. addr is the starting address of the display ram to which the lead display is directed.

4.8.1. V.24 INTERFACE RS232C

(According to INTERNATIONAL STANDARD ISO 2110-1980(E))

PIN#	MNEMONIC	FORTH MNEMONIC	DESCRIPTION
1	PG		Protection Ground
2	SD		Send Data
3	RD		Receive Data
4	RTS	IRS	Request to Send
5	CTS	ICS	Clear to Send
6	DSR	IDM	Data Set Ready
7	SG		Signal Ground
8	CD	IRR	Carrier Detect
9 & 10		RESERVED FOR NATIONAL USE	
11			Unassigned
12	SCD		Secondary Carrier Detect
13	SCTS		Secondary Clear to Send
14	SSD		Secondary Send Data
15	SCT		Tx Clock From DCE
16	SRD		Secondary Receive Data
17	SCR		Rx Clock from DCE
18			Unassigned
19	SRTS		Secondary Request to Send
20	DTR	ITR	Data Terminal Ready
21	SQD	ISQ	Signal Quality
22	RI	IIC	Ring Indicate
23	DRS	ISR	Data Signal Rate Select
24	SCTE		Tx Clock to DCE
25			Unassigned

4.8.2. V.11 INTERFACE

(According to INTERNATIONAL STANDARD ISP 4903-1980 (E))

Where balanced circuits are concerned, the associated pairs are designated "A" and "B".

PIN#	MNEMONIC	FORTH MNEMONIC	DESCRIPTION
1	PG		Protection Ground
2	T(A)		Transmit
3	C(A)	IC	Control
4	R(A)	IR	Receive
5	I(A)	II	Indicate
6	S(A)		Signal Element Timing
7	B(A) or F(A)		Byte Timing (X.21) Frame Start Indication (X20)
8	SG		Signal Ground
9	T(B)		Transmit
10	C(B)	IC	Control
11	R(B)	IR	Receive
12	I(B)	II	Indicate
13	S(B)		Signal Element Timing
14	B(B) or F(B)		Byte Timing (X.21) Frame Start Indication (X20)
15	RESERVED FOR FUTURE INTERNATIONAL USE		

4.8.3. V.35 INTERFACE (34 PINS)

(According to INTERNATIONAL STANDARD ISO 2593-1973 (E))

N = Pin number permanently reserved for national use.

F = Pin number permanently reserved for future international standard and should not be used for national use.

Subscripts indicate pins which may be associated to form pairs; for example F1 on pins K and M form a pair.

PIN#	MNEMONIC	FORTH MNEMONIC	DESCRIPTION
A	PG		Protective Ground
B	SG		Signal Ground
C	RTS	IRS	Request to Send
D	CTS	ICS	Clear to Send
E	DSR	IDM	Data Set Ready
F	CD	IRR	Carrier Detect
H	DTR	ITR	Data Terminal Ready
J	RI	IIC	Ring Indicate
K			F1
L			F2
M			F1
N			F2
R	RD		Received Data A-Wire
T	RD		Received Data B-Wire
V	SCR		Receiver Signal Element Timing A-Wire
X	SCR		Receiver Signal Element Timing B-Wire
Y	SCT		Transmitter Signal Element Timing A-Wire (From DCE)
AA	SCT		Transmitter Signal Element Timing B-Wire (From DCE)
P	SD		Transmitted Data A-Wire
S	SD		Transmitted Data B-Wire
U	SCTE		Transmitter Signal Element Timing A-Wire (To DCE)
W	SCTE		Transmitter Signal Element Timing B-Wire (To DCE)
Z			F3
BB			F3
CC			F4
DD			F5
EE			F4
FF			F5
HH			N1
JJ			N2
KK			N1
LL			N2
MM			F
NN			F

4.8.4. V.36 INTERFACE

4.8.4.1. 37 PIN CONNECTOR

(According to INTERNATIONAL STANDARD ISO 4902-1980 (E))

A, B and C indicate the associated interchange points as designated in CCITT recommendations V.10 and V.11. The B designation applies only when a V.11 generator is used and the C designation applies only when a V.10 generator is used.

PIN#	MNEMONIC	FORTH MNEMONIC	DESCRIPTION
1	PG or SG		Protective Ground or Signal Ground
2	SI	ISI	Data Signal Rate Select
3&21			Reserved for national Use
4	SD		Transmitted Data (A)
22	SD		Transmitted Data (B or C)
5	ST		Transmit Signal Element Timing from DCE (A)
23	ST		Transmit Signal Element Timing from DCE (B or C)
6	RD		Received Data (A)
24	RD		Received Data (B or C)
7	RS	IRS	Request to Send (A)
25	RS	IRS	Request to Send (B or C)
8	RT		Receiver Signal Element Timing from DCE (A)
26	RT		Receiver Signal Element Timing from DCE (B or C)
9	CS	ICS	Ready for Sending (A)
27	CS	ICS	Ready for Sending (B or C)
10	LL	ILL	Local Loopback
11	DM	IDM	Data Set Ready (A)
29	DM	IDM	Data Set Ready (B or C)
12	TR	ITR	Data Terminal Ready (A)
30	TR	ITR	Data Terminal Ready (B or C)
13	RR	IRR	Data Channel Received Line Signal Detector (A)
31	RR	IRR	Data Channel Received Line Signal Detector (B or C)
14	RL	IRL	Remote Loopback
15	IC	IIC	Calling Indicator
16	SR or SF	ISR	Data Signal Rate Select to DCE or Select Tx Frequency to DCE
17	TT		Transmit Signal Element Timing from DTE (A)
35	TT		Transmit Signal Element Timing from DTE (B or C)
18	TM	ITM	Test Indicator
19	SG		Signal Ground
20	RC		DCE Circuit Ground
28			Terminal Available for Service
32	SS	ISS	Select Standby
33	SQ	ISQ	Signal Quality Detector
34	NS	INS	New carrier Disable CD Response Time to DCE
36	SB	ISB	Response to SS (Pin #32)
37	SC		DTE Circuit Ground

4.9. BEEPER CONTROL

The beeper control commands access a variable: **BEEP_DUR** which determines the on and off times of the beeper. The user may modify the value as defaulted in the operating system.

BEEP	(--) Produces one beep.
BEEP_DUR	(-- addr) Returns the address of the variable used to set beeper on/off times.
BEEP_ON	(--) Turns the beeper on.
BEEP_OFF	(--) Turns the beeper off. One beep is guaranteed if this command is executed quickly after the BEEP_ON .
BEEP_TONE	(-- addr) Returns the address of the variables used to set the tone of the beeper.
TONE	(tone --) Sets the tone of the beeper without affecting the value saved in BEEP_TONE . Note: This word communicates directly with the hardware.

4.10. TIMER

START_TIMER	(timer_no/time --) Starts a timer which, if not stopped (or restarted), will time out and provide a timeout indication. Where timer_no is a numeric value assigned to the timer to allow identification upon timeout or for the purpose of stopping the timer. The number is any arbitrary value chosen by the user in the range 1 to 128; and where time is the number of one-tenth second increments the timer is to run. Values between .1 and 429254250 seconds are valid. The resolution is ~30 msec.
START_TIMER_20	(timer_no/time --) Same as START_TIMER , but time is the number of one-fiftieth second increments the timer is to run.
STOP_TIMER	(timer_no --) Stops the specified timer. Where timer_no is the numeric value assigned in the START_TIMER command.
SLEEP	(#_of_one_tenth_of_a_second --) Puts FORTH into sleep for the designated number of one-tenth of a second. The largest unit it can sleep is $2^{32}-1$.

4.11. TIME

SET_TIME (YY/MO/DD/HH/MI/SS --)

Sets the real time clock

Where Sets the real YY=year,MO=month,DD=day,HH=hour,MI=minute,SS=sec. Each entry is a 2 digit number, (except for the year which must be entered as a 4 digit number eg. 1985) 0<=HH<24, and entries are separated from each other with spaces.

SET_TIME sets the value in **TBASEYR** to the year that is specified on the stack.

The year that is specified in the SET_TIME command is the base year for any subsequent timestamp. This base year will also be saved when a file is saved, so that all the timestamps in the file are associated with the file's base year.

GET_TIME (-- SS/MI/HH/DD/MO/YY)

Reads the real time clock and returns the values on the computational stack.

The format of the time will be as described in the SET_TIME command except that the order is reversed.

TBASEYR (-- base_year)

This word will return the base year set by SET_TIME or the system time at bootup if SET_TIME has not been used.

BASEYEAR (base_year --)

BASEYEAR sets the base year that is used for subsequent timestamp conversions. This would be the value returned by TBASEYR or FDBASEYR.

CLOCK_DSP ({YES|NO} --)

Turn on/off the clock display on the status line.

GET_TS (timestamp_addr --)

Puts a 48 bit timestamp into the timestamp address. Use the base year set by BASEYEAR as the starting year.

TSTAMP_ADD (addr_ts_1/addr_ts_2/addr_for_sum --)

Adds the 48 bit timestamp value in addr_ts_1 to the 48 bit timestamp value in addr_ts_2. The resulting 48 bit value is stored in addr_for_sum. Assumes the designated timestamps have used the same base year.

TSTAMP_COMP (addr_ts_1/addr_ts_2 -- =?)

Compares the 48 bit timestamp1 value in addr_ts_1 to the 48 bit timestamp2 value in addr_ts_2. Assumes the designated timestamps have used the same base year.

Returns: -1 if timestamp1 is less than timestamp2
 0 if timestamp1 equals timestamp2
 +1 if timestamp1 is greater than timestamp2

TSTAMP_SUB (addr_ts_1/addr_ts_2/addr_for_diff -- flag)

Subtracts the 48 bit timestamp2 value in addr_ts_2 from the 48 bit timestamp1 value in addr_ts_1. Assumes the designated timestamps have used the same base year.

Returns: -1 if timestamp2 is greater than timestamp1

GET_TSTAMP (timestamp_addr -- μ ss/mss/ss/mi/hh/dd/mo/yy)
Converts the timestamp to output format. Uses the base year set by
BASEYEAR as the starting year.

The timestamp is returned on the stack as follows (top to bottom)
YEAR, MONTH, DAY, HOUR, MINUTE, SECOND, MILLISECOND,
MICROSECOND.

GET_TSTAMP_MICRO (timestamp_addr -- μ ss/ss/mi/hh/dd/mo/yy)
Converts the timestamp to output format. Uses the base year set by
BASEYEAR as the starting year.

The timestamp is returned on the stack as follows (top to bottom):
YEAR, MONTH, DAY, HOUR, MINUTE, SECOND, MICROSECOND.

GET_TSTAMP_MILLI (timestamp_addr -- mss/ss/mi/hh/dd/mo/yy)
Converts the timestamp to output format. Uses the base year set by
BASEYEAR as the starting year.

The timestamp is returned on the stack as follows (top to bottom):
YEAR, MONTH, DAY, HOUR, MINUTE, SECOND, MILLISECOND.

SORT_ON ({YES|NO} --)
Switches the time stamp sort algorithm ON or OFF. The default value is
YES.

The GET_TS, TSTAMP_ADD, TSTAMP_COMP and TSTAMP_SUB words can be used in conjunction with the GET_TSTAMP, GET_TSTAMP_MICRO and GET_TSTAMP_MILLI words. The timestamp words also are used with the Start Timestamp and End Timestamp returned by UNPACK_DATA.

4.12. DISK INTERFACE

For all disk interface commands (except write operations), FORTH execution is suspended until the command has been carried out. In all cases, the error status of the previous command may be placed on the computational stack using the **FDSTATUS** command. 0 indicates success, a positive number is a data indication, and a negative number is an error indication.

4.12.1. DISK DRIVE SELECTION

Selection of drive will be according to FORTH conventions. **DR0** and **DR1** are standard FORTH commands used to select floppy drives 0 and 1 respectively.

CUR_DEV (-- device_number)
Returns the value identifying the current drive on the computational stack.

The following values will be returned:

0	=	DR0
1	=	DR1
256	=	WD0
257	=	WD1
.		
.		
.		
263	=	WD7
264	=	WD10
.		
.		
.		
271	=	WD17

SET_CUR_DEV (device_number --)
Sets the current selected disk drives using the devices numbers. See also **CUR_DEV**.

4.12.2. FILE ACCESS CODES

To allow opening of files for various access modes, four access modes have been defined:

RE (-- value)
Read only.

WR (-- value)
Write only.

REWR (-- value)
Read/Write.

WWR (-- value)
Write with wraparound, i.e. overwrite.

4.12.3. FILE CLOSE CODES

TRUNCATE (-- d)
Returns the value for file truncation.

NO_TRUNCATE (-- d)
Returns the value for no file truncation.

4.12.4. DISK FILENAMES

Filenames may be up to 14 characters in length and may contain the following characters: A-Z, a-z, 0-9, _ , . , +

Note: Filename leaves the address of the filename on the stack, therefore another filename entry mechanism could be utilized, provided the address is left on the stack.

4.12.5. FILE DESCRIPTOR

FDES (-- addr)
FDES is a variable that contains the file descriptor of the last disk file opened.

CREATE_FILE (filename/access/size/type -- file_desc)
Creates a file with the specified name, size and access mode. The file is also opened automatically, and if opened successfully, a file descriptor (16 bit numeric value) will be returned on the computational stack. If the operation is unsuccessful, the value on the computational stack is undefined. The file will be created on the drive presently selected by the commands **DR0**, **DR1**, **WD0**, **WD1**, ... , **WD7**, **WD10**, **WD11**, ..., **WD17**.
Where filename is the name of the file and is entered in the form " FILENAME" and access is the access format ie. WR - write or WWR - write with wrap around and SIZE is the number of 512 byte blocks and TYPE is one of the FILE TYPE CODES.

FILE TYPE CODES

0xA = EXEC (Operating System)
0x1 = DATA
0x2 = OBJ
0x4 = SOURCE
0x10 = REMOTE SOURCE
0x20 = USER
0x40 = CONFIGURATION
0x80 = TEST MANAGER BINARY

CREATE_FILE uses the value in **TBASEYR** as a base year for creating the file and to set the value in **FDBASEYR**.

- OPEN_FILE** (filename/access -- file_desc)
Opens a file for the access mode specified. Successful opening will return a file descriptor on the computational stack. If the operation is unsuccessful, the value on the computational stack is undefined. This file descriptor will be used by the close, read, write and seek commands.
- Where filename is the name of the file and is entered in the form, " FILENAME" and access is the access mode, i.e. **WR** – write, **WWR** – write with wraparound, **RE** – read, or **REWR** – read/write.
- OPEN_FILE** sets the value in **FDBASEYR** to the base year associated with that file.
- CLOSE_ALL** (--)
Closes all open files. Files which are opened for writing will be truncated, and empty files will be removed.
- CLOSE_FILE** (truncate_flag/file_esc --)
Closes the file. The command may truncate the file or not by passing the appropriate parameter. If the file is empty, it will be discarded.
- Where truncate flag may be **TRUNCATE** or **NO_TRUNCATE** and where file desc. is the value returned by the open sequence.
- WRITE_BLOCKS** (file_desc/buffer_addr/no_of_blocks --)
Writes the specified number of blocks onto the disk. Where file_desc is the value returned by the open sequence and buffer_addr. is the name of the buffer containing the data to be written. Buffer addr. is a variable or other pointer. No_of_blocks is the number of 512 byte blocks that are to be written.
- NOTE:** The last block will also consist of 512 bytes.
- CLOSE** (--)
Closes the file whose descriptor is in the variable **FDES**. This will be the last file which was opened.
- READ_BLOCKS** (file_desc/buffer_addr/no_of_blocks --)
Reads the specified number of blocks from the disk. Where file_desc is the value returned by the open sequence and buffer addr. is the name of the buffer into which the data will be read. No. of blocks is the number of 512 byte blocks that are to be read.
- NOTE:** The last read operation will also read a complete block of 512 bytes, even if only a partial block had been written.
- NOTE:** If the buffer does not have sufficient space to accommodate the file, the following memory area will be overwritten.
- SEEK** (file_desc/offset --)
Moves to an absolute offset within a file. Where file_desc is the value returned by the open sequence and offset is the number of blocks (absolute) to offset into the file starting with 0.
- REMOVE** (filename --)
Removes the filename from the directory. **REMOVE** will not remove a system file. Where filename is of the form " FILENAME".

-
- RENAME** (*old_filename/new_filename --*)
 Renames a file.
 Where old filename and new filename are both of the form
 " *FILENAME*".
- FORMAT** (*--*)
 Formats the currently selected disk. (floppy only)
- MAKE_FS** (*volume --*)
 Makes a file system on the currently selected disk. Where volume is the name of the volume.
- WRTPRT_FS** ({ *YES | NO* } *--*)
 Sets the software write protect flag for the filesystem on the current selected drive. For example:
 DR0 YES WRTPRT_FS
 write-protects the filesystem on *DR0*. While,
 WD0 NO WRTPRT_FS
 removes the software write protect flag for the filesystem on *WD0*.
- COPY** (*source/dest --*)
 Copies the file or files from the source diskette to the destination diskette. Whenever complete diskettes are copied, an automatic compression is performed. Where source is the source filename. This may include the drive designation eg. *DR0:*. It may also be a regular expression; however, under these circumstances the filename will be duplicated on the destination diskette. Dest. is the destination filename.
- DIR** (*filename --*)
 Gets a listing of the directory. This is displayed on the screen. Where filename is the filename or regular expression (including wildcards i.e. *, ? and [.]).
- DIRL** (*filename --*)
 Gets a listing of the directory (long form). This is displayed on the screen. Where filename is the filename or regular expression (including wildcards i.e. *, ? and [.]).
- LOCATE_FILE** (" *filename*" *--* { *0 | 1* })
 Locates the specified file from the filesystem on *DR0*, *DR1*, *WD0*, *WD1*, *WD2*, *WD3*, *WD4*, *WD5*, *WD6*, *WD7*, *WD10*, *WD11*, *WD12*, *WD13*, *WD14*, *WD15*, *WD16*, *WD17*. If the file exists on the filesystem that it is searching, then the search is terminated and the corresponding drive will stay selected and the value one is returned on the stack. If the file does not exist in any of the available filesystems, then *DR0* will be selected and the value zero is returned on the stack. Temporary files are not locatable by **LOCATE_FILE**.

LOCATE_FILE sets the value in **FDBASEYR** to the base year associated with the file.

 Warning: **FDSIZE** and **FDTYPE** will be set if the file exists.
- FDBASEYR** (*-- base year*)
 Returns the base year that is associated with the most recently opened, located or created file.
-

FDSIZE (-- file size)

Returns the size of the last opened, created, or closed file in 512 byte blocks. (FDSIZE) is the memory address that contains the file size.

FDSTATUS (-- status)

Returns the status of the previous disk command on the computational stack. 0 indicates no error, and a negative number is an error indication. These values are:

- 3 File does not exist.
- 4 Unrecoverable physical I/O error! Disk not formatted?
- 5 Nonexistent I/O device.
- 6 Bad file descriptor.
- 7 File access prohibited.
- 8 File already exists.
- 9 Error in Disk I/O Function.
- 10 Invalid File Name.
- 11 End of file.
- 12 No space in filesystem.
- 13 Too Many Files Opened.
- 14 Cannot create a new filesystem on the disk because there are files opened in the existing filesystem (File System Busy).
- 15 Floppy disk not loaded.
- 16 No 'RTE' file system.
- 17 Disk is write protected.
- 18 Can't copy file to itself.
- 19 Bad source device.
- 20 Bad destination device.
- 21 No match for Wild Cards.
- 22 Difference in compared files.
- 23 File system is write protected.

FDTYPE (-- file type)

Returns the file type of the last opened, created, closed, or located file. The file type is the same as the one used by **CREATE_FILE**.

COMPARE (source/dest --)

COMPARE compares the file or files from the source diskette to the destination diskette. Where source is the source filename. This may include the drive designation, eg. DR0:. It may also be a regular expression; however, under these circumstances the filename will be duplicated on the destination diskette. Destination is the destination filename.

F_REMOVE (filename --)

Removes the designated file with force. This word may be used to remove a system file. The syntax is " **FILENAME** " **F_REMOVE**.

RPT_D_ERR (status --)

Reports on the screen, the status of the last disk operation. All negative numbers returned by **FDSTATUS** are converted to a clear text message for interpretation by the user.

RDIR	(filename/dir_type -- addr_of_dir_buffer/no_of_lines) Returns the address of the buffer the directory has been loaded into and the number of lines in the directory. 0 is short directory type and 1 is long directory type
REPORT	(--) Displays, on screen, the status of last disk operation.
DISK_BUF	(-- addr) DISK_BUF is a 1024 byte array used by some disk operations as a read/write buffer.
(NFLOPPY)	(-- mask) Returns the mask for the number of floppy drives that exist in the system. Each bit in the mask represents a floppy drive. Bit 0 corresponds to drive 0 and bit 1 corresponds to drive 1. The floppy drive exists if the corresponding bit value is 1.
NFLOPPY	(-- number_of_floppy_drives) Returns the number of floppy drives that exist in the system.
(NWINC)	(-- mask) Returns the mask for the number of logical Winchester drives that exist in the system. Each bit in the mask represents a logical Winchester drive. Bit 0 corresponds to logical drive 0, bit 1 corresponds to logical drive 1, and bit 2 corresponds to logical drive 2 and so on. The logical drive exists if the corresponding bit value is 1.
NWINC	(-- number_of_logica_winchester_drive) Returns the number of logical Winchester drives that exist in the system.

4.12.6. HARD DISK SUPPORT

This section applies to units that have one or more Winchester Drives installed.

4.12.6.1. ORGANIZATION OF A WINCHESTER DISK

A Winchester disk is divided into logical disks. A logical disk is either an entire physical disk or part of a physical disk. The possible number of logical disks on a physical disk are:

1. Only one logical disk on the physical disk, i.e. the size of the logical disk is the same as that of the physical disk.
2. More than one and less than nine logical disks on a physical disk, i.e. the sum of the size of the logical disks is less than or equal to the size of the physical disk and no logical disks can overlap.

The names for the disks are:

WD0, WD1, ... , WD7 for logical disks on drive 0
WD10, WD11, ... , WD17 for logical disks on drive 1

After a Winchester disk is partitioned into logical disk(s), all DIO commands except **FORMAT**, operate on the designated logical disk.

For example:

```
WD0 " WD0" MAKE_FS
```

makes a filesystem called **WD0** on logical disk 0 of drive 0.

4.12.6.2. COMMANDS

The words **WD0**, **WD1**, **WD2**, **WD3**, **WD4**, **WD5**, **WD6** and **WD7** select the designated logical disk on Winchester drive 0. The words **WD10**, **WD11**, **WD12**, **WD13**, **WD14**, **WD15**, **WD16** and **WD17** select the designated logical disk on Winchester drive 1. If the designated logical disk does not exist, then an error message is printed on the FORTH window and the current selected drive stays selected. The syntaxes are:

WD0 (--)	WD10 (--)
WD1 (--)	WD11 (--)
WD2 (--)	WD12 (--)
WD3 (--)	WD13 (--)
WD4 (--)	WD14 (--)
WD5 (--)	WD15 (--)
WD6 (--)	WD16 (--)
WD7 (--)	WD17 (--)

PARTITIONWD (*LS0/LS1/.../LSn/#LS --*)

Logically partitions the selected Winchester disk.

Where, *LS0* is the size of logical disk 0 in 512 byte block. *LS1* is the size of the logical disk 1 in 512 byte block. *LSn* is the size of the logical disk *n* in 512 byte block. *#LS* is the number of logical disk's size placed on the stack.

For example:

```
2000 3000 2 PARTITIONWD
```

Creates logical disk 0, which occupies the first 2000 blocks and logical disk 1, which occupies the next 3000 blocks on the current selected Winchester drive. If the Winchester drive has over 5000 blocks, then logical disk 2 will be created and occupies the rest of the disk. On the other hand, if the Winchester drive has less than 5000 blocks, then the size of logical disk 1 becomes the size of the Winchester minues the size of logical disk 0. If the size of the Winchester drive is less than 2000 blocks, then logical drive 1 is not created, and the size of logical drive 0 is equal to that of the physical drive.

```
WD2 0 PARTITIONWD
```

Creates logical disk 0 on Winchester drive 0 which occupies the entire physical drive.

FORMATWD (*LS0/LS1/.../LSn/#LSI --*)

Formats and logically partitions the selected Winchester disk. The parameters passed to **FORMATWD** are used for logical disk partitioning.

Where, *LS0* is the size of logical disk 0 in 512 byte block. *LS1* is the size of the logical disk 1 in 512 byte block. *LSn* is the size of the logical disk *n* in 512 byte block. *#LS* is the number of logical disk's size placed on the stack.

See the examples in **PARTITIONWD** for the partitioning strategy.

4.12.7. DATA RECORDING CONTROL

Data recording is managed by the data stream merger (DSM) function. Data is recorded one track at a time, i.e. 10 blocks for floppy and 16 blocks for harddisk of 512 bytes/block. To accommodate this, the DSM function maintains three buffers, each large enough to hold a track full of data.

The **DREC_ON** command creates a data file with the specified size and wraparound indication. The file descriptor and current device for the recording file are then mailed to the DSM function.

Data to be recorded will be sent to the DSM function from the originating FORTH machine using **RE_ROUTE**.

Note: The IPC buffer is mailed to the DSM function rather than the data buffer.

DREC_OFF (--)
Turns data recording off.

DREC_ON (filename/size/access --)
Creates the data recording file and sends the necessary information to the DSM function. Where filename is the name assigned to the data file size is the number of 512 byte blocks. ACCESS is the access mode:
WR – for write
WWR – for write with wraparound

(DSTATUS) (-- addr)
Returns the address of variable that contains the Data Recording Status.

Note: These functions are not part of the dictionary but are included in the applications.

DSM_RECORD? (--)
Returns the recording status of the DSM. This word returns 0 if an application is already recording else -1. Each application should use this word to check if it have to do recording file handling (removing the old recording file creating and opening the new file etc.) The syntax is :

4.12.8. DISK PLAYBACK CONTROL

The following words are designed to aid playback of any data recorded on disk.

Note: All lengths and counts are in bytes.

D_INIT (" filename"/addr --)
Opens the designated file and initializes disk playback procedure if the file can be opened successfully. The status is stored in (DSTATUS) where: "filename" is the designated data file. Address is the address of the buffer which holds the data from the disk.

D_NEWEST (-- { length/addr | null })
Returns the length and address of the newest block of data from the disk playback file. A zero is returned if the file is empty or an error is encountered. (DSTATUS) contains the status of this operation.

D_NEXT	(-- { length/addr null }) Returns the length and address of the next block of data from the disk playback file. A zero is returned if the file is empty or an error is encountered. (DSTATUS) contains the status of this operation.
D_OLDEST	(-- { length/addr null }) Returns the length and address of the oldest block of data from the disk playback file. A zero is returned if the file is empty or an error is encountered. (DSTATUS) contains the status of this operation.
D_PREVIOUS	(-- { length/addr null }) Returns the length and address of the previous block of data from the disk playback file. A zero is returned if the file is empty or an error is encountered. (DSTATUS) contains the status of this operation.
D_STOP	(--) Closes the data file and terminates data playback from disk.

4.13. DISPLAY MANAGER

The Display Manager provides the application programmer a flexible, robust, easy to use method of screen management. It is possible to create multiple windows in which textual output may be directed. For example, a three line window may be created for the status area, a 14 line window used for protocol decoding, and a three line window for the FORTH command line. All of these windows operate independently of each other, that is, each will scroll in its own area and not disturb the contents of any other window. Windows may also overlap one another. Once again, information entering a window will not disturb the contents of any superimposed window.

There are many additional parameters which may be used for window manipulation. This section will explain the creation, characteristics of, and use of display windows.

WINDOW CREATION AND POSITIONING

The programmer may create a window using the FORTH word **BUY_WINDOW**. This will return a number on the stack known as a **window descriptor**. It is a number which uniquely identifies a window. There are 16 windows in all, two of which are reserved for system use. Any attempt to "BUY" more than 14 windows will result in a return value of "0". This indicates there are no more windows to allocate. Coincidentally, zero also happens to be the window descriptor of the main FORTH window (the one visible on power up).

BUY_WINDOW (start row/no. of rows -- wd)
Will attempt to create a new window. It is passed the starting row of the window on the physical screen (0..22) and the number of rows the window is to contain (1..22). Upon return, the stack will contain the new window descriptor or zero indicating that there are no more available windows.

BUY_WINDOW does not display the window, nor does it enable it for output. It simply creates the necessary structures in memory for its eventual use.

SELL_WINDOW (window descriptor --)
Sells the designated window back to the system. This is the complement of **BUY_WINDOW**.

SHOW_WINDOW (wd --)

Will make the indicated window visible on the screen. This is useful for "uncovering" a window hidden under other windows.

POP_WINDOW (wd --)

Is the function equivalent of performing first a **SHOW_WINDOW** and then an **OPEN_WINDOW** on a specific window.

INIT_DISPLAY (--)

Initializes display manager of that FORTH machine to startup condition with no windows active.

SHIFT_WINDOW (n --)

Will shift the open window vertically on the display screen by +/- "n" units up or down. (See **LOCK_WINDOW** and **UNLOCK_WINDOW**).

DROP_WINDOW (window_descriptor --)

Removes the designated window from the display.

WINDOW CONFIGURATION COMMANDS

Once a window is created, several options may be selected. These options relate to whether the window will scroll when full or wrap around, and whether the direction is up or down. When a window is first created, it is defaulted to scrolling in the downward direction.

WSCROLL (--)

Makes the currently selected window into a scrolling window. The direction of output will remain unchanged.

WRAP (--)

Makes the currently selected window into a wrap around window. The direction of output will remain unchanged.

WUP (--)

Sets the output direction of the active window to up.

WDOWN (--)

Sets the output direction of the active window to down.

CURSOR_OFF (--)

Disables the display of the block cursor.

CURSOR_ON (--)

Enables the display of a reverse video, block cursor within the window. The cursor will indicate the position of the next character to be output.

AUTO_CLEAR_OFF (--)

The opposite of **AUTO_CLEAR_ON**.

AUTO_CLEAR_ON (--)

This word informs the display manager to clear the following line after a carriage return is printed. This is the default

OUTPUT COMMANDS

The Display Manager provides a complete set of FORTH words to allow the application programmer to present data in many different formats. This flexibility ensures that sophisticated visual displays may be created with a minimum of programming effort.

- OPEN_WINDOW** (wd --)
Will enable all further output from the window commands to be placed in this window.
- CLOSE_WINDOW** (--)
Will cancel an **OPEN_WINDOW** and revert the active window to the previously opened window.
- THERE** (row/col --)
Will position the cursor of the active window to co-ordinate(row,col). If either row or column is out of range with respect to the active window, it will be adjusted to fit.
- WHERE** (-- row/col)
Returns the cursor position of the active window.
- W.TYPE** (string/count --)
Is very similar to **TYPE**, **STYPE**, etc. It will output count characters of string to the active window. The mode of output will be in accordance with the options described above. Under no circumstances will the textual output overrun the boundaries of the active window. The attributes used will be those last set with the **SET_ATTR** command.
- W.TYPE_A** (string/count/attr --)
Is very similar to **W.TYPE** except that the string which is output will be printed in the color and character set specified by attribute.
- W.TYPE_RC** (string/count/row/col --)
Will print "string" at location (row,col).
- W."** (--)
Behaves like **."** except that the output string will be sent to the active window. The attributes used will be those last set with the **SET_ATTR** command.
- W.MOVE** (wstring/wcount --)
Will take a string of "words" (16 bit values) and output them to the display. This output will take the attribute values from the first byte in every word.

Where "wcount" is the number of words in the string.
- W.** (d --)
Behaves like **.** except that the output digits will be sent to the active window. The attributes used will be those last set with the **SET_ATTR** command.
- (W. ") (--)
Runtime part for **W."** to display text in current display window.
- W.H** (d --)
Behaves like **.H** except that the output hex digits will be sent to the active window.

W.R	(n/columns --) (-- ascii_value) Format : 'a, leaves ascii code of "a"
W.SPACES	(n --) Display n Spaces on the window
WCR	(--) Will perform a carriage return operation on the currently active window. This is done in accordance with the current settings of the scroll, direction and the auto clear switches. That is, it may either scroll up or down, or wrap up or down.
CLEAR_TEXT	(--) Will clear all text (i.e. characters) from the currently active window. All attributes will be left in place and the cursor will be placed at row 0, column 0 (ie top left corner of the window).
CLEAR_ROW	(row --) Will clear the text from only one row of the active window and will place the cursor at column 0 of that row.
CLEAN_WINDOW	(--) Erases the text and the paint of the current window.
PAINT	(attr --) Will change the entire active window to attribute. The text in the window will remain unaffected. This may also be used to change the character set of a range of characters. i.e. change ASCII to HEX.
PAINT_FIELD	(row/col/length/attr --) Will paint a number of positions starting at a given location in the specified color and character set. This word will paint to a maximum of one screen row.
PAINT_ROW	(row/attr --) Will paint a specified row.

MISCELLANEOUS COMMANDS

SAVE_SCREEN	(--) Will save an image of the current screen's window positions.
RESTORE_SCREEN	(--) Complements SAVE_SCREEN . RESTORE_SCREEN will take the saved screen image and return it to the display screen.
NO_SCROLL_DISABLE	(--) Disables the NOSCROLL key on the keyboard. This should be done on the initialization of an application program.
NO_SCROLL_ENABLE	(--) Enables the NOSCROLL key on the keyboard.

LOCK_WINDOW (window --)

Locks the specified window so that no other windows will be allowed to overwrite the screen space occupied by it. That is, no other windows will be able to be popped over top of a locked window. Also, this window cannot be shifted.

UNLOCK_WINDOW (window --)

Reverses the effect of a **LOCK_WINDOW** .

{FORTH-W} (-- addr)

Variable may be used to re-route regular FORTH output from the FORTH window (window 0) to another window of your choice.

For example:

```
1 {FORTH-W} !
" *" DIRL      <-- request directory
2 {FORTH-W} !
VLIST          <-- request VLIST
0 {FORTH-W} !  <-- restore normal FORTH output
```

Here a dictionary listing will appear in window number one, while the vocabulary listing will appear in window number two.

SYSTEM PARAMETERS AND FUNCTIONS

NOTE: The use of these words are not recommended without knowledge of the structure and operating of the Display Manager internal mechanisms.

W-ADDRESS (-- addr)

Will return the beginning RAM address of the active window. This is useful for using words such as **M.TYPE**, **M.MOVE**, etc. within a display manager environment.

Wcb (-- addr)

Will return the address of the window control block of currently active window.

Wcb_table (-- addr)

Will return the address of window control block allocation table. It contains pointers to window control blocks.

Window# (-- addr)

Returns the address of the active window stack. The currently active window may be found by ANDing the number at this location with 0xf.

FirstRow (-- row)

Will return the starting physical row number of the currently active window.

STATUS-W (-- wd)

This returns the window descriptor for the status window (top line of the display) so that this window may be made the active window. This window is not permanently locked. .

Cols	(-- n) Cols returns the number of columns in a window.
DISTABLE	(-- addr) DISTABLE returns the address of the display table. Warning: Handle with care while playing with the contents of the display table.
FCOLORS	(-- addr) Returns the address of a 4 byte array which contains colors for the function key tags. Byte 0 corresponds to the color used by the bottom row; byte 1 corresponds to the color used by the middle row; byte 2 corresponds to the color used by the top row; and byte 3 is not used and should always contain 0.
Fcol	(-- column) Returns the first column of the current window.
Rows	(-- rows) Returns the number of rows in the current window.
Rowtable	(-- addr) Returns the address of the row table in the window control block. Warning: Handle with care when playing with the contents in the row table.
Status	(-- addr) Returns the address of the status word in the window control block for the current window. Warning: Handle with care while playing with the contents of the status word.
WD#	(-- window_descriptor) Returns the window descriptor for the current window.
WSTACK	(-- addr) Returns the address of the window stack. Warning: Handle with care while playing with the contents of the window stack.
POP_ROW	(row --) Pops the designated row from the window stack and puts it on the display.

4.14. DISPLAY

These Commands are used for displaying messages to screen.

Alternative to the **M.** commands are **W** for screen display.

RST_INQ	(--) Clears the terminal input queue (active only when the application is managing the display, i.e. 'smart' mode).
---------	--

SET_ATTR (attr --)

Sets the display attribute to be used in the output command(s) following.
Where attr is the display attribute according to the following list (these may be OR'ed):

Color/Attribute	Foreground	Background
White	WHI_FG (-- value)	WHI_BG (-- value)
Red	RED_FG (-- value)	RED_BG (-- value)
Green	GRN_FG (-- value)	GRN_BG (-- value)
Blue	BLU_FG (-- value)	BLU_BG (-- value)
Yellow	YEL_FG (-- value)	YEL_BG (-- value)
Cyan	CYA_FG (-- value)	CYA_BG (-- value)
Magenta	MAG_FG (-- value)	MAG_BG (-- value)
Black		BLK_BG (-- value)
Reverse Video	REV_VIDEO (-- value)	
ASCII char. set	ASCII (-- value)	
EBCDIC char. set	EBCDIC (-- value)	
HEX char. set	HEXSET (-- value)	
TTX char. set	TTX (-- value)	
JS8 char. set	JS8 (-- value)	

Note: If the attribute is set to -1, the colour will return to that of the last PAINT command.

GET_ATTR (-- value)

Returns the present attribute setting on the stack.

SAVE_DSPP (--)

Saves cursor and display position and color attribute condition. These will be used for the next **RESTORE_DSPP**.

RESTORE_DSPP (--)

Restores cursor and display position and color attribute to the condition that existed when the last **SAVE_DSPP** was executed.

SAVE_ATTR (-- addr)

SAVE_ATTR is a variable that is used to temporarily hold the display color attribute byte.

STYPE (start_addr/no_of_char --)

Displays a text string on the screen. The command functions in a manner identical to the **TYPE** command in FORTH except that the entire command has been written in assembly language to provide higher speed operation. The attributes used will be those last set with the **SET_ATTR** command.

Where start_addr is the address of the text string and no_of_char is the number of characters to be displayed.

STYPE_A (start_addr/no_of_char/attr --)

Similar to **STYPE** except that attributes are passed as a parameter.

Where start_addr is the address of the text string and no_of_char is the number of characters to be displayed, and where attr is the display attribute as returned by the constants as defined in **SET_ATTR**.

4.15. M. DISPLAY ROUTINES

The following instructions support mapped display memory. **POINTER-M** is the pointer to the start of the display line and **ATTR-M** is the attribute byte stored in the attribute portion of each memory location.

M.TYPE	(offset/string_addr/count --) M.TYPE will output the character string at string_addr with count into the word string pointed to by POINTER-M with offset giving the starting position. The high byte will be loaded with the value in ATTR-M to give color and character set selections.
M.MOVE	(string_addr/offset --) Output a character string which has the count in its first byte to the offset past POINTER-M in a word format with ATTR-M as high byte.
M.	(value/offset --) Output a decimal value converted to ASCII to the offset past POINTER-M in a word format with ATTR-M as high byte.
M.HH	(value/offset --) Output a HEX value low 4 bits converted to ASCII to the offset past POINTER-M in a word format with ATTR-M as high byte.
M.HB	(value/offset --) Output a HEX byte value converted to ASCII to the offset past POINTER-M in a word format with ATTR-M as high byte.
HEX_LINE	(source_addr/count/offset -- updated_source_addr) Outputs one line in HEX bytes from address for count bytes. The address is incremented by count and is returned on the stack. Note: HEX_LINE truncates the line if the output string is longer than the line.
POINTER-M	(-- addr) POINTER-M is a variable which contains the display line address used by M.TYPE , M.MOVE , M. , M.HH , M.HB and HEX_LINE .
ATTR-M	(-- addr) ATTR-M is a variable which contains the color and character set attribute to be used by M.TYPE , M.MOVE , M. , M.HH , M.HB and HEX_LINE .
THIS-RX	(-- addr) A variable containing the current receiver array. Used by SET_SCREEN_POINTER .
SET_SCREEN_POINTER	(--) Sets POINTER-M using THIS-RX array pointer. Utilized by the Formatted Monitor Programs.

4.16. CURSOR POSITIONING

CUR_A	(row/col --) Positions cursor at row and column position specified. Where row and col. are row and column number at which the cursor is to be positioned. Permissible values are as follows: row 1 - n col 0 - 79
--------------	---

CUR_P (-- row/col)

Reports the cursor position. The position of the cursor within the scene will be returned on the stack in the same order as the **CUR_A** command.

4.17. FUNCTION KEY MANAGER

The Function Key Manager provides the application programmer a powerful, yet simple, tool for creating and modifying the function key portion of the user interface. Through the use of the function key manager, the responsibility for all aspects of screen and keyboard display have been removed from the programmer and automated. Thus, by isolating the programmer from actual physical management, program development time is reduced through the faster creation of user interfaces.

The initialization of the Function Key Manager is done through the word **INIT_FKEYS**. It is passed, as a parameter, the number of lines of function keys needed. This can be in the range of one to three lines of keys.

There are two main concepts involved in the Function Key Manager. These are namely, what a function key label **says**, and what pressing that key actually **does**.

In most cases, the pressing of a key should execute the FORTH word which appears on its label. For example, suppose we wish to have UF3 (unshifted function key 3) labelled "HALT" and also execute the FORTH word "HALT" when pressed. The following sequence can be used:

```

: HALT      ( FORTH definition of word )
    ...
    ...
    ...
;           ( end of definition of HALT )
'" HALT " UF3 DISPLAY_KEY

```

This would display the text "HALT" on the screen and remember that the word **HALT** should be executed. Note the use of "'" (tick-quote), this is a signal to **DISPLAY_KEY** to compile both the text and the action of the word "HALT".

Other times, it is desirable to have the action of the key to do something that the text does not say. It may be just a different FORTH word or several different FORTH words. For example, to create a key labelled "STOP" but to have it execute "BOTH_RX RX_OFF" the following sequence may be used:

```

" STOP " UF3 DISPLAY_KEY      ( display the text )

MAKE_KEY UF3
    1 FROZEN !                ( store value in variable )
    BOTH_RX RX_OFF ;          ( declare the action )

```

In fact, it is possible, through **MAKE_KEY**, to make a key execute several different actions. For example:

```

" FREEZE " SF5 DISPLAY_KEY    ( display the text )

MAKE_KEY SF5
    BOTH_RX RX_OFF            ( turn off receivers )
    ENABLE_CAPT ;              ( enable capture buffer )

```


WORD DESCRIPTIONS

- INIT_FKEYS** (*n* --)
Initializes the Function Key Manager.
- Where the number "*n*" specifies the number of lines to display on the bottom of the screen.
- FKM-LINES** (-- *addr*)
Returns the address of variable. This variable contains the number "*n*" which represents how many function key lines are active.
- FKM-SCENE** (-- *addr*)
Variable contains the RAM address of the beginning display memory used for the function key manager.
- "** (-- *addr/fk#*)
This word is used in conjunction with **DISPLAY_KEY** to assign the action of a word to that key as well as show that word. For example:
- ```
' " MENU" SF1 DISPLAY_KEY
```
- This assigns the action **MENU** to the **SF1** key and also displays **MENU**.
- ""** ( -- *addr/fk#* )  
This word is similar to **"** except the word displayed doesn't have to exist. For example:
- ```
"" HELLO" UF1 DISPLAY_KEY
```
- This displays **HELLO** and assigns the action **" HELLO" EXECUTE_WORD** to the **UF1** key (See **EXECUTE_WORD**).
- DISPLAY_KEY** (*string addr/fk#*)
Used with **"** or **'** or **""** to display a key tag on the specified function key. For example:
- ```
" DISK-ON" CF8 DISPLAY_KEY
```
- DISPLAY\_KEY\_A** ( *string addr/fk#/color* -- )  
Is exactly like **DISPLAY\_KEY** but will display the text in the provided color. For example:
- ```
" DISK-ON" CF8 BLU_BG DISPLAY_KEY_A
```
- HILITE_KEY** (*fk#* --)
Will make the entire label area for "*fk#*" blue background (**BLU_BG**).
- UNLITE_KEY** (*fk#* --)
Complements **HILITE_KEY**, and returns the key text to its default color.
- HILITE_FIELD** (*fk#/begin/end* --)
Will highlight certain characters within a particular key label. The beginning and end range from 1 to 9, while all other text will be set to its default color. **UNLITE_KEY** may be used to cancel the effects of this word.

PAINT_KEY (fk# /begin/end/color --)
Will color a specific portion of a key label in the color of your choice.

KILL_KEY (fk# --)
Removes both the label text and the action associated with a particular key.

FKM-WINDOW (-- address of the window descriptor)
Returns the address of the window descriptor for the function keys display.

MAKE_KEY (--)
May be used to assign multiple actions to a particular key. This in no way will affect the label text of a key. MAKE_KEY must be terminated with either the end of a word definition, i.e. ';' or ';AND'. For example:

```
: TEST_WORD
  MAKE_KEY UF1
    STACK
    0 flag !
    X21_decode

    ; <-- this semi-colon terminates the MAKE_KEY

< or >

: TEST_WORD2
  MAKE_KEY UF1
    STACK
    0 flag !
    X21_decode

    ;AND <-- terminate the first MAKE_KEY

  MAKE_KEY UF6
    1 flag !
    X25_decode

    ; <-- terminate the second MAKE_KEY and colon
```

EXECUTE_KEY (fk# --)
Takes the function key code passed and executes the action (if any) associated with that key. If no action has been specified for that key, nothing will happen.

BEGIN_KEY_SEQ (--)
Starts the definition of an n-element function key sequence. This provides an automatic method for implementing a function which assumes multiple values. NOTE: Must be followed by a fk#. For example, the X.25 Monitor functions: **REPORT-SHORT**, **REPORT-HEX**, **REPORT-COMPLETE**. By using **BEGIN_KEY_SEQ**, we can easily see how to implement this:

```

BEGIN_KEY_SEQ UF3                                <-- start a key sequence

    " REP-SHORT " UF3 DISPLAY_KEY <-- done initially
;NEXT
    " REP-HEX "    UF3 DISPLAY_KEY <-- next action for key
;NEXT
    " REP-COMP "   UF3 DISPLAY_KEY <-- next action for key etc.
;NEXT

REPEAT_KEY_SEQ

```

;NEXT (--)

Serves to delimit action boundaries in a key sequence. It means that when the key is pressed, the action on the next press will be the FORTH words up to the next ;NEXT. **Note:** Used by the BEGIN_KEY_SEQ construct.

REPEAT_KEY_SEQ (--)

Serves to close off a user-defined key sequence. When the action preceeding the REPEAT_KEY_SEQ is executed, the first action will be the next action in the sequence. It may be noted, that the above example will always start out in the **REP-SHORT** mode. i.e. it has no memory of its former state once reloaded. To provide memory, the following solution is advised:

```

0 VARIABLE REPORT-FLAG                          <-- variable holding state

BEGIN_KEY_SEQ UF3                                <-- start a key sequence
    REPORT-FLAG @                                <-- fetch variable

    CASE 0
    {
        " REP-SHORT " UF3 DISPLAY_KEY
        ;NEXT 1 REPORT-FLAG !
    }
    CASE 1
    {
        " REP-HEX " UF3 DISPLAY_KEY
        ;NEXT 2 REPORT-FLAG !
    }
    CASE 2
    {
        " REP-COMP " UF3 DISPLAY_KEY
        ;NEXT 0 REPORT-FLAG !
    }

    REPEAT_KEY_SEQ

```

This next example illustrates how you may use the IF-ELSE-THEN structure for a yes-no decision.

```

O VARIABLE FLAG

BEGIN_KEY_SEQ UF6

    " ON/OFF " UF6 DISPLAY_KEY
    FLAG @
    IF
        UF6 1 2 HILITE_KEY
        ;NEXT 0 FLAG !
    ELSE
        UF6 4 6 HILITE_KEY
        ;NEXT 1 FLAG !
    THEN

REPEAT_KEY_SEQ

```

4.18. KEYBOARD INPUT

Keyboard entry is on a character by character basis. Further processing of the data stream from the keyboard is handled by the front end.

BUY_KEYS (port/key group --)
 Allows the FORTH task to notify TCF that it will assume ownership of certain groups of keys.
 Where port is 0 = LOCAL, 1 = REMOTE and where key group may be one of the following (numeric values for the parameters are shown in parentheses):

KBD	(-- value)	- entire keyboard excluding function and cursor keys
CURSOR	(-- value)	- cursor motion keys
UF	(-- value)	- unshifted function keys
SF	(-- value)	- shifted function keys
CF	(-- value)	- control function keys

The key groups may be OR'ed.

INP_TERM (char --)
 Terminal entry termination character. If character is equal to 1 then a **STR_IND** is generated for every character entered. Then use **KEY** to extract the character and return it on the stack. Where char. may be any character value except 'BS' and 'LF'. The default is CR (carriage return).

4.19. ERROR RECOVERY & ERROR PROTECTION

Error protection and recovery is usually needed when interpreting input from the keyboard or compiling a file from within a program. Sometimes, it is necessary to nest the error protection or create protective shells. There are two methods available in the system, namely **SET_RETURN** and **SHELL{ and }SHELL** pairs. The first method does not provide any nesting capability, but the latter one does. The two mechanisms **are not** designed to use in conjunction with each other.

4.19.1. SET_RETURN

SET_RETURN (--)

Enables return to second word following it, after execution of next word; if this next word calls **TAKE_RETURN** or **TAKE_ERROR_RETURN** or calls any word that uses these. This is normally used when calling **INTERPRET** in user program. It allows **INTERPRET** to return an error flag to the user program without halting execution.

Usage: SET_RETURN INTERPRET IF ." OK"
ELSE ." ERROR"
THEN

TAKE_RETURN (-- 1)

If **SET_RETURN** has been used, **TAKE_RETURN** pushes a one (1) on the stack and returns to the second word following the **SET_RETURN**. Else, **TAKE_RETURN** does nothing.

TAKE_ERROR_RETURN (-- 0)

If **SET_RETURN** has been used, **TAKE_ERROR_RETURN** pushes a zero on the stack and returns to the second word following the **SET_RETURN**. Else, **TAKE_ERROR_RETURN** does nothing.

?RETURN_SET (-- flag)

This flag is set by **SET_RETURN** and cleared by **TAKE_ERROR_RETURN** or **TAKE_RETURN**.

(SET_RETURN) (--)

Run-time part of **SET_RETURN** .

(TAKE_ERROR_RETURN) (-- 0)

Executed by **INTERPRET** if an error occurs when a **SET_RETURN** has been done.

(TAKE_RETURN) (-- 1)

Executed by **INTERPRET** if there are not any errors when a **SET_RETURN** has been done.

4.19.2. SHELL{ AND }SHELL

SHELL{ (--)

Initiates a protective shell for the word following it, which allow nested error protection and recovery for the word that follows. **SHELL{** must be used in conjunction with **}SHELL**.

WARNING: No more than 7 shells can be nested at any one time. This cannot be used with **SET_RETURN** at the same time.

}SHELL (-- flag)

Terminates the shell and returns a flag indicating if an error was encountered in executing the word preceding it.

For example:

```
SHELL{INTERPRET}SHELL
IF
    ."OK"
ELSE
    ."ERROR"
ENDIF
```

- EXIT_SHELL** (--)
Exits from the shell. If at some point, in a protected word, processing can no longer continue, then this word may be used to exit from the protecting shell.
- SHSP!** (--)
Resets the shell stack pointer to point to the bottom of the shell stack.
- S** (-- n)
Copies the element from the top of the shell stack and pushes it onto the parameter stack. Analogous to **R**.
- >S** (n --)
Takes an item from the parameter stack and pushes it onto the shell stack. Analogous to **>R** and just about as lethal if misused.
- S>** (-- n)
Pops an item from the shell stack and pushes it onto the parameter stack. Analogous to **R>** and just about as lethal if misused.

4.20. CAPTURE BUFFER CONTROL

There is one capture buffer on each FORTH machine. The structure of the data is described under DATA INDICATION BUFFER STRUCTURE: (Interface lead transition & Receive data buffer structure). Retrieved blocks can be passed through **UNPACK_DATA**.

Several routines have been provided for the management of the capture buffer.

Note: All lengths and counts are in bytes.

The following routines are supported:

CONFIGURATION

- INIT_CAPTURE** (wrap flag --)
Initialize the capture buffer indicating how many times wraparound should be allowed. If the parameter is 0 no wraparounds will be done, otherwise wraparounds will be allowed.
- Where wrap flag specifies wraparound control.
- CB_VAR_INIT** (id --)
Used to initialize either capture or disk playback scrolling. id=0 for capture and 1 for disk playback.

C-B	(-- addr) Variable which contains the address of the memory to be used for capture buffer.
C-B-SIZE	(-- addr) Variable which contains the size in bytes of the capture buffer.

STORING

CAPTURE	(count/addr -- captured?) Puts the data at the address into the capture buffer. If the capture buffer is full (i.e. has wrapped around as many times as specified) the indication will be NO, otherwise it will be YES. If no limit on wraparound was specified, the indication will always be YES. Where count is the length of the data to be captured and addr. is the address of the data.
----------------	---

RETRIEVAL

OLDEST	(-- count/addr) Returns the address of the oldest block in the capture buffer. If there are no blocks at all, NULL is returned instead of an address.
NEWEST	(-- count/addr) Returns the address of the newest block in the capture buffer. If there are no blocks at all, NULL is returned instead of an address.
NEXT	(-- count/addr) Returns the address of the next block in the capture buffer with respect to the last request. If there are no newer blocks, or no blocks at all, NULL is returned instead of an address.
PREVIOUS	(-- count/addr) Returns the address of the previous block in the capture buffer with respect to the last request. If there are no newer blocks, or no blocks at all, NULL is returned instead of an address.

4.21. PRINTER PORT

The **PRINT** set of commands allow sending async data out of the 'To Printer' port. This port supports async ASCII serial RS232C printers.

The Printer Configuration is done by the commands **BPCH**, **STOP_BITS**, **ODD_PAR**, **EVEN_PAR**, **TX_SPEED**, **RX_SPEED**, **RX_CONTROL**, **CHAR/LINE**, **LINES/PAGE**, **PRINTER_EOL**, **PR_A/H_TABLE**, **PRINT_MODE**, **PRINTER CONFIG**. These are detailed in Section \$R\$ Configuration.

PRINT	(addr/count/byte word flag -- status) Spools character string to the printer port.
--------------	--

Where 1 is the flag for a byte string and 0 is the flag for a word string (Display Screen format with character in low byte). The **PRINT** command will return a status of 0 for a success and either **FTHEBUF** or **FTHEMAIL** for a failure (See **EXTRACT_ERROR**).

PRINT_W (addr/count/byte word flag --)
Sends character string to the printer port and waits until transmission is completed.

Where YES refers to a byte string and NO refers to a word string.

PRINT_EOL (--)
Prints the End of Line sequence as defined with the **CR_EOL PRINTER_EOL** or **LFCR_EOL PRINTER_EOL** commands.

Note: If word string is used in **PRINT** or **PRINT_W**, then the high byte of the word is treated as an attribute for that character and that character is interpreted and printed according to its attribute. See also **SET_ATTR**.

4.22. REMOTE MODEM PORT

The **REMOTE** set of commands allows sending async data in and out of the 'To MODEM' port. This port supports async serial RS232C Modems.

The Remote Configuration is done by the commands **BPCH**, **STOP_BITS**, **ODD_PAR**, **EVEN_PAR**, **NO_PAR**, **TX_SPEED**, **RX_SPEED**, **REMOTE CONFIG**. These are detailed in Section \$R\$ Configuration.

REMOTE_OUT (addr/count/byte word flag -- status)
Spools character string to the remote modem port.

Where 1 is the flag for a byte string and 0 is the flag for a word string (Display Screen format with character in low byte). The **REMOTE_OUT** command will return a status of 0 for a success and either **FTHEBUF** or **FTHEMAIL** for a failure (See **EXTRACT_ERROR**).

REMOTE_OUT_W (addrcount/byte word flag --)
Sends character string to the printer port and waits until transmission has completed.

Where 1 is the flag for a byte string and 0 is the flag for a word string.

LEAD CONTROL

The remote leads may be controlled using the **TURN_ON** and **TURN_OFF** commands. Constants have been defined for DTR (**RMT_DTR**) and RTS (**RMT_RS**).

RMT_DTR (-- value)
RMT_DTR is the value used for **TURN_ON** and **TURN_OFF** to control the **REMOTE** port DTR lead.

RMT_RS (-- value)
RMT_RS is the value used for **TURN_ON** and **TURN_OFF** to control the **REMOTE** port RTS lead.

4.23. CONFIGURATION

It is possible to configure each of the test port directions individually. This is accomplished by specifying PORT ID in the configuration commands where PORT ID is one of the test port designators according to the list:

```

PORT1_A
PORT1_B or PORT1 for both directions
PORT2_A
PORT2_B or PORT2 for both directions
PORT3_A
PORT3_B or PORT3 for both directions
PORT4_A
PORT4_B or PORT4 for both directions
PORT5_A
PORT5_B or PORT5 for both directions
PORT6_A
PORT6_B or PORT6 for both directions
PRINTER - not classified as a test port
REMOTE  - not classified as a test port

```

Note: The various configuration commands set up the configuration buffers only. The actual hardware is configured using the CONFIG command.

4.23.1. INTERFACE

I/F_MODE (port/mode --)
 Establishes the interface mode for the specified port. The value entered is stored in a variable **I_MODE**.
 Where mode may be:
MON - monitor
TODCE - K1197 emulates a terminal interface connected to a DCE
TODTE - K1197 emulates a modem interface connected to a DTE

eg. PORT1 MON I/F_MODE sets PORT1 as monitor (two receivers)

I_MODE (-- addr)
I_MODE is a variable containing the current interface lead mode.

GET_I/F (port -- physical_interface_type)
 Determines the type of interface selected for the specified port. One of the following values will be placed on the computation stack.

```

0 = None
1 = V.24/V.28
2 = V.11
3 = V.36
4 = V.35

```

I/F_TYPE**(port/physical interface type --)**

Sets the physical interface type for the designated port. The interface types are: V28, V28MIL, V35, V36, V11.

V28 (-- n) – returns the constant that corresponds to the V28 type interface.

V11 (-- n)

V35 (-- n)

V36 (-- n)

V28MIL (-- n)

FAST_M**(port --)**

Configures the System FAST Recording Mode.

All incoming data are routed to the DSM and recorded on disk (DSM must be activated with DREC_ON !). The DSM will independently route the DATA to the application. Lost of DATA is avoided because the application event processing does not influence the main data stream.

Note: The recording media must be fast enough to save all incoming data. For Floppy Disk the max. data rate will be 20kBytes/sec for Winchester 50kBytes/sec if no additional Memory Extension card is installed else 100kBytes/sec (this data rate includes additional information stored on disk: timestamps, block sequence count, block forward and backward pointer etc.) **Note:** Maximum data rates are for 1 Erlang of traffic at 64 kbps.

NORM_M**(port --)**

Ceases the effect of FAST_M and places the system in normal mode.

PROTOCOL**(port/type --)**

Establishes the protocol for a specific test port. Where port is one of the test port designators as described above and type is the procedure according to the following list (numeric values are given in parentheses):

- | | | |
|--------|--------------|------------------------------------|
| (0) | BOP | – bit oriented procedure (primary) |
| (1) | | reserved for BOP (secondary) |
| (2) | | reserved |
| (3) | COP | – character oriented procedure |
| (4) | BSC_E | – bisync procedure with EBCDIC |
| (5) | BSC_A | – bisync procedure with ASCII |
| (6) | ASYN | – asynchronous |
| (7) | ISOC | – isochronous |
| (0x13) | X.21 | – X.21 procedure |

Note: Whenever this command is executed, the configuration buffer is initialized to the default values as shown in the following table.

 PROTOCOL DEFAULT VALUES

	BOP	COP	BSC_E	BSC_A	ASYNCR	ISOC
INTERFACE MODE	MONITOR	MONITOR	MONITOR	MONITOR	MONITOR	MONITOR
CLOCKING	STANDARD	STANDARD	STANDARD	STANDARD	-	STANDARD
BITS PER CHAR.	8	8	8	7	8	
STOP BITS	-	-	-	-	1	
PARITY	NONE	NONE	NONE	ODD	NONE	
SYNC CHARACTER	0X7E	0X16	0X32	0X16	-	
INTERFRAME FILL	0X7E	MARK	MARK	MARK	-	
REST IDLE	-	MARK	MARK	MARK	MARK	
SYNC. RESET	-	0XFF	-	-	-	
DCD CONTROL	OFF	OFF	OFF	OFF	-	
CRC	CRC-CCITT	NONE	CRC-16	VRC/LRC	-	
LEADING PAD	-	0XFF	0XFF	0XFF	-	
SPEED	2400	2400	2400	2400	1200	
STRIP SYNC.	ON	ON	ON	ON	-	
NRZ/NRZI	NRZ	NRZ	NRZ	NRZ	-	
EOF CHARACTER	-	-	-	-	20	
COUNT						
EOF TIMEOUT	-	-	-	-	17 MSEC	
FAST MODE	NO	NO	NO	NO	NO	NO

4.23.2. CLOCKING SELECTION

STD_CLK (port --)

Selects the modem provided clocks as transmit and/or receive clocks.

EXT_CLK (port --)

Selects the external clock source eg. pin 24 for RS232-C.

NRZ (port --)

None return to zero coding. This is the same as STD_CLK.

NRZI (port --)

None return to zero inverted coding. This selection is used for some SDLC and SNA lines.

NRZIC (port --)

None return to zero inverted coding with clock extraction from the data stream.

SPEED (port/value --)

Specifies the operating speed in bits per second. The speeds supported are 50, 75, 110, 134, 150, 200, 300, 1200, 1800, 2000, 2400, 3600, 4800, 7200, 9600, 14400, 16000, 19200, 38400, 48000, 56000, 64000, 72000, 128000 and 256000.

Where value is a constant or variable containing the bit rate as a decimal number.

SPEED_TABLE (-- addr)

Array with the valid speeds allowed by speed.

4.23.3. USART SETUP

Allows usart characteristics to be entered

BPCH	(port/value --) Specifies the bits per character. Where value is the number of bits per character and may be entered as 5, 6, 7 or 8.
STOP_BITS	(port/value --) Specifies the number of stop bits for asynchronous operation. Where value is the number of stop bits and may be entered as 1, 1.5 or 2. Note that 1.5 is entered as 15.
EVEN_PAR	(port --) Specifies even parity.
ODD_PAR	(port --) Specifies odd parity.
NO_PAR	(port --) No parity to be calculated.
SYNC_CHAR	(port/value --) Specifies the sync character. Where CHAR is the sync character in HEX.
SYNC_RESET	(port/value --) Specifies the character which will cause the receiver to start a new sync search (COP only). Where value is the character in HEX.
DCD_CONTROL	(port/id --) Specifies whether the DATA CARRIER DETECT lead going to the off condition is to resynchronize the receiver. Where id may be: (0) OFF – do not resynchronize receiver (1) ON – resynchronize receiver
IF_FILL	(port/value --) Specifies the interframe fill condition. Where value may be one of the following: (0) SPC (1) MRK (2) SYNC – the sync character is used as specified in the SYNC_CHAR command.
REST_IDLE	(port/value --) Specifies the idle condition before transmission starts. Where value may be: (0) SPC (1) MRK

LEADING_PAD	(port/value --) Specifies the leading pad character for BSC and COP procedures. Where value is a HEX entry for the pad.
STRIP_SYNC	(port/value --) Controls whether sync characters are to be stripped in the receive operation (COP only). Where value may be: (0) OFF (1) ON
BCC	(port/value --) Specifies BCC type. Where value may be: (0) CRC_CCITT (1) CRC_16 (4) NO_BCC (2) VRC/LRC
CRC_CCITT	(-- value) This constant is used to identify the CRC type used by CRC_CALC or BCC to be CCITT CRC calculation.
CRC_16	(-- value) This constant is used to identify the CRC type used by CRC_CALC or BCC to be CRC 16 calculation.
VRC/LRC	(-- value) VRC/LRC BCC type
EOF_COUNT	(port/value --) Configures receiver to force an end of frame condition when the specified number of characters has been received. (ASYNC and COP) Where value specifies the number of characters. NOTE: In FAST-MODE Configuration for BOP EOF_COUNT specifies the number of Data-Packet forcing an end of frame condition.
ASYNC_TIME (Async Only)	(port/time --) Configures receiver to force an end of frame condition when the specified time has elapsed since receipt of the last character. Where time specifies the number of milliseconds NOTE: In FAST-MODE Configuration for BOP ASYNC_TIME specifies a TIMEOUT value in 0.5 μ s units forcing an end of frame condition.

4.23.4. ASYNC PORT CONTROL

The following section applies to PRINTER and REMOTE port configuration. The commands **BPCH**, **STOP_BITS** and **ODD_PAR**, **EVEN_PAR** and **NO_PAR** are used to configure the UART. Split speed operation is permitted and to support this two speed selection commands are available.

Note: The **CONFIG** command must be executed to set up the hardware. The configuration buffer is defaulted to 1200 bps, 8 bpch, no parity, 80 characters per line, 60 lines per page, DTR and XON/XOFF both active.

The **PRINTER_MODE** default selection is **CHAR_DISPLAY** and the **PRINTER_EOL** default selection is **LFCR_EOL**.

TX_SPEED	(port/value --) Sets the baud rate for the transmitter
	Where value may be one of 50, 75, 110, 134.5, 150, 200, 300, 1200, 1800, 2000, 2400, 4800, 9600, 19.2K.
RX_SPEED	(port/value --) Sets the baud rate for the receiver.
	Where value is one of the list shown under TX_SPEED .
TX_CONTROL	(port/id --) Sets TRANSMITTER control mode.
	Where id may be:
	OFF
	DTR - DATA TERMINAL READY control
	XOF - XON/XOFF control
	DTR&XOF - both of above
CHARS/LINE	(port/value --) Sets the number of characters which constitute a line. After this number of characters is sent on the same line, the system will insert an end of line.
LINES/PAGE	(port/value --) Sets the number of lines which constitute a page, i.e. to determine when form feed is to be generated.
PR_A/H_TABLE	(addr --) Transfers the address of the ASCII/HEX table for TRA_PRINT_MODE to the system.
PRINTER_EOL	(id --) Sets a variable (PRINTER_EOL) which determines the sequence generated for an end of line condition. Where id may be:
	NONE_EOL - not generated automatically
	CR_EOL - carriage return is generated
	LFCR_EOL - line feed is generated, followed by carriage return.

EOL (-- value)
EOL returns the printer end of line character currently used.

PRINTER_MODE (mode --)
Sets the variable (**PRINTER_MODE**) which determines the display format.

Where mode may be:

- | | |
|---------------------|--|
| CHAR_DISPLAY | - outputs all non-printable characters in format <AB> where A,B are the two nibbles represented in hex notation. |
| RAW_DISPLAY | - outputs all characters without doing any range checking. |
| HEX_DISPLAY | - outputs all hex characters in format <AB> where A,B are the two nibbles represented in hex notation. |
| TRA_DISPLAY | - outputs dependent on the configuration in the ASCII/HEX table. HEX characters are printed in format <AB> where A,B are the two nibbles represented in HEXotation. If HEX notation is not selected, the charater is printed in ASCII. |

RMT_BLK_MODE (--)
RMT_BLK_MODE puts the operation of the remote (MODEM) port in block mode. In this mode a **RMT_STR_IND** event is produced only after a carriage return is received from the remote port.

RMT_CHR_MODE (fifo_addr/size_of_fifo --)
RMT_CHR_MODE puts the operation of the remote (MODEM) port in the character mode. The user has to supply a buffer, which is used as a fifo to store the characters received from the remote port. The **RMT_STR_IND** event is produced only when the fifo goes from empty to non-empty. **RMTGETC** can be used to extract a character from the fifo.

4.23.5. CONFIGURATION BUFFERS FOR CONFIGURATION INFORMATION

These buffers are sent to the send/receive functions which are active for the specified interface mode or to the terminal console function if the PRINTER or REMOTE ports are selected.

```
PORT1_ACBUF      ( -- addr )
PORT1_BCBUF      ( -- addr )
PORT2_ACBUF      ( -- addr )
PORT2_BCBUF      ( -- addr )
PORT3_ACBUF      ( -- addr )
PORT3_BCBUF      ( -- addr )
PORT4_ACBUF      ( -- addr )
PORT4_BCBUF      ( -- addr )
```

Returns the address of the configuration buffer for PORT 1A, PORT 1B, PORT 2A, PORT 2B, PORT 3A, PORT 3B, PORT 4A and PORT 4B, respectively.

```
REMOTE_CBUF      ( -- addr )
                  Address of the remote async port configuration buffer.
```

```
PRINTER_CBUF     ( -- addr )
                  Address of the printer port configuration buffer.
```

```
.IBUF             ( conf_buf -- )
                  Displays the 32 word configuration buffer values. The configuration
                  buffers are named:
                  PORT1_ACBUF, PORT1_BCBUF, PORT2_ACBUF, PORT2_BCBUF,
                  PRINTER_CBUF, REMOTE_CBUF .
```

STRUCTURE OF THE CONFIGURATION BUFFER

WORD 1-6:	System Use (IPC-Buffer header)
WORD 7:	Interface mode
WORD 8:	Protocol
	0 = BOP
	1 = reserved
	2 = reserved
	3 = COP
	4 = BSC EBCDIC
	5 = BSC ASCII
	6 = ASYNC
	7 = ISOC
WORD 9:	Clocking
	0 = standard
	1 = external
WORD 10:	Bits per character
	0 = 5
	1 = 6
	2 = 7
	3 = 8
WORD 11:	Stop bits
	0 = 1
	1 = 1.5
	2 = 2
WORD 12:	Parity
	0 = NONE
	1 = ODD
	2 = EVEN
WORD 13:	Sync character
	Hex character
WORD 14:	Interframe fill
	0 = Space
	1 = Mark
	2 = Sync
WORD 15:	Idle character
WORD 16:	Rest idle
	0 = Space
	1 = Mark
WORD 17:	LSByte Sync reset (COP only)
	Hex character
	MSByte DCD control
	0 = OFF
	1 = ON
WORD 18:	Crc
	0 = CRC-CCITT
	1 = CRC-16
	2 = VRC/LRC
	4 = NO_CRC
WORD 19:	Leading pad (HEX character)

WORD 20:	Speed
	0 = 50
	1 = 75
	2 = 110
	3 = 134.5 (may be entered as 134 or 135)
	4 = 150
	5 = 200
	6 = 300
	7 = 1200 (default value for Printer and Remote Ports)
	8 = 1800
	9 = 2000
	10 = 2400 (default value for Test Ports)
	11 = 3600
	12 = 4800
	13 = 7200
	14 = 9600
	15 = 14400
	16 = 16000
	17 = 19200
	18 = 38400
	19 = 48000
	20 = 56000
	21 = 64000
	22 = 128000
	23 = 256000
WORD 21:	Strip sync
	0 = No
	1 = Yes
WORD 22:	NRZ/NRZI selection
	0 = NRZ
	1 = NRZI
WORD 23:	Transmit control (PRINTER/REMOTE) – FAST MODE Selection
	0 = OFF (Test Ports)
	1 = XOF
	2 = DTR
	3 = DTR&XOF
WORD 24–25:	Address of ASCII/HEX-table for TRA_PRINT_MODE / Starting address of FIFO
WORD 26:	Size of FIFO (internal)
WORD 27:	Character count for forced EOF (async and COP only), max. Frame count for BOP-FAST MODE
WORD 28:	Timeout value for forced EOF (async only), TIMEOUT value for BOP-FAST MODE in 0.5 μ s units
WORD 29:	Receive speed for PRINTER/REMOTE, Interface character type
WORD 30:	Characters per line (PRINTER)
WORD 31:	Lines per page (PRINTER)
WORD 32:	Terminal type (PRINTER/REMOTE)

CONF_BUF_ASYNC	(-- addr) Default configuration buffer ASYNC.
CONF_BUF_BOP	(-- addr) Default configuration buffer BOP (HDLC and SDLC).
CONF_BUF_BSCA	(-- addr) Default configuration buffer BISYNC ASCII.
CONF_BUF_BSCE	(-- addr) Default configuration buffer BISYNC EBCDIC.
CONF_BUF_COP	(-- addr) Default configuration buffer CHARACTER ORIENTED PROTOCOL.
CONF_BUF_ISOC	(-- addr) Default configuration buffer ISOC.
CONF_BUF_CCS#7	(-- addr) Default configuration buffer for CCS#7 monitor and simulator.
CONFIG	(port --) Performs usart configuration for the selected port.
(NFECPCBD)	(-- mask) Returns the mask for the number of front end communication processor boards that exist in the system. Each bit in the mask represents an FECPC board. Bit 0 corresponds to the FECPC 1, bit 1 corresponds to FECPC 2 and bit 2 corresponds to FECPC 3. The FECPC board exists if the corresponding bit value is 1.
NFECPCBD	(-- number_of_FECPC_boards) Returns the number of FECPC boards that exist in the system.

4.24. CONFIGURATION OF ISDN INTERFACES

4.24.1. CONFIGURATION OF THE BASIC ACCESS INTERFACE (BAI)

4.24.1.1. INTRODUCTION

Due to the Hardware concept of the new BAI (Basic Access Interface) board, there is a variety of features, which can be selected easily using the FORTH commands implemented specially for this purpose. These Hardware features together with an enhanced Software basis enable the new BAI board to function as very reliable, powerfull and flexible measurement instrument in almost every application for ISDN Protocol testing. The necessary FORTH commands are described briefly and where an example seemed to be of use, it is done. The operating sytem includes driver processes for S0 and Up₀ Interfaces. The configuration for the chosen Interface (S₀ or Up₀) is done automatically by an identification procedure for Layer1 called during the system boot up. As result, the same Software (i.e. System and Application) may be loaded for both Interfaces. The following FORTH commands, integrated in the FORTH base dictionary, allow the configuration of the BAI board in K1197. These commands and their belonging constants are local and therefore available only on CPU7 and CPU8, i.e. first and second BAI CPU boards. All commands are invoked with the proper parameters, i.e. portid etc, as described in the command syntax line. Please note, that all configuration mode commands are activated after the FORTH command CONFIG is invoked. The numbers given in brackets in some commands are the corresponding values for parameters, which should be entered instead of their names.

4.24.1.2. CONFIGURATION MODE REALTED COMMANDS

SET_PRIORITY (portid / priority --)

specifies the access priority for D channel on the S-Bus, where portid has to be PORT_BRID and priority is one of:

- LPRIO - allows an Activation Request of Layer1 with low priority 8 on the S-Bus,
- HPRIO - allows an Activation Request of Layer1 with high priority 10 on the S-Bus.

default: LPRIO

Note: This command should be used in TE mode in connection with a S0 interface.

BUS_MODE (portid / mode --)

specifies the access mode for Layer1 device, where portid has to be PORT_BRID and mode is one of:

- P2P - configures the S0 connector module for Point to Point operation,
- SHORT_BUS - configures the S0 connector module for Short-Bus mode operation,
- EXTENDED_BUS - configures the S0 connector module for Extended-Bus mode operation. This mode is available only in NT mode.

default: SHORT_BUS

Note: This command can be used only in connection with a S0 Interface.

TERM_RES (portid / mode --)
specifies, whether the termination resistor on the S0 connector module should be switched to 100 Ohm or to High Impedance, where portid has to be PORT_BRID and mode is one of:

- ON - sets the termination resistors of the S0 connector module to 100 Ohm,
- OFF - sets the termination resistors of the S0 connector module to High impedance.

default: ON

Note: *This command can be used only in connection with a S0 Interface.*

POWER_FEEDING (portid / mode --)
specifies, which power feeding source should be used, where portid has to be PORT_BRID and mode is one of:

- OFF - switches any internal power feeding off, so that an external power feeding is possible,
- ON - switches normal internal power feeding on using the on board DC/DC Converter,
- (2) INVERS - switches invers internal power feeding on using the on board DC/DC Converter.

Note: *The internal power feeding should be used only when there is no external power source connected to the connector module. This command should be used only in NT mode.*

default: OFF

CLOCK_SOURCE (portid / mode --)
specifies the clock source for interface and attached peripherie, where portid has to be PORT_BRID and mode is one of:

- (0) INTERNAL - clock source is the internal 15,36 MHz oscillator,
- (1) EXTERNAL - clock source is the an external Clock generator, which has to be connected via a BNC connector to the S0 connector module,
- (2) SLAVE - clock source is a second Interface in K1197, e.g. S2M board or another S0 interface.

Note: *Setting the clock source to external mode should be used only after the connection with the clock generator has been established.*

default: INTERNAL

4.24.1.3. LAYER1 AND LAYER2 RELATED COMMANDS

PH:ACTIVATE (portid --)

activates the S-bus, where portid has to be PORT_BRIDA or PORT_BRIDB due to the interface mode NT or TE.

Note: This command can't be used in monitor mode.

PH:DEACTIVATE (portid --)

deactivates the S-bus, where portid has to be PORT_BRIDA or PORT_BRIDB due to the interface mode NT or TE.

Note: This command can't be used in TE and monitor mode.

MPH_ERROR_RESPONSE (portid --)

sets the Layer1 device into the deactivated state, where portid has to be PORT_BRIDB. This command should be used whenever no activation is possible or K1197 does not respond properly to the messages coming from Layer1.

Note: This command should be used only in TE mode.

STATE? (portid -- / -- state)

returns the state of Layer1 device due to the interface mode, where portid has to be PORT_BRIDA, PORT_BRIDB or PORT_BRID due to the interface mode NT, TE or MONITOR and the returned state code is one of:

S0 Interface:

TE:

- (0) SENSING,
- (1) ACTIVATED,
- (2) DEACTIVATED,
- (5) WAITING_FOR_SIGNAL.
- (6) IDENTIFYING_INPUT,
- (7) SYNCHRONIZED,
- (9) LOST_FRAMING.

NT:

- (1) ACTIVE,
- (2) DEACTIVATED,
- (3) PENDING_ACTIVATION,
- (4) PENDING_DEACTIVATION,

MON:

- returns the same state codes as NT.

Note: The returned state is left on the FORTH stack. The numbers given in brackets refer to the states of Layer1 device (SBC PEB 2080). These do not accord to the state numbers given in CCITT I.430. The figures given above can only be used, when a S0 Interface is used.

Up0 Interface:

TE:

- (1) ACTIVATED,
- (2) DEACTIVATED,
- (3) Temporary Interstate.

NT:

- (1) ACTIVE,
- (2) DEACTIVATED,
- (3) Temporary Interstate.

MON:

~ returns the same state codes as NT.

Note: *The Temporary Interstate means that the Up0 Line is not synchronized and it will come to a defined active or inactive state after a while.*

C/I_MON

(portid / mode --)

turns the Layer1 monitor on or off, so that any Layer1 state changes is reported via an c/i-indication to the attached FORTH application. The portid has to be PORT_BRIDA, PORT_BRIDB or PORT_BRID due to the interface mode NT, TE or MONITOR and mode is one of:

ON ~ enables the C/I monitor,

OFF ~ disables the C/I monitor.

default: OFF

Note: *In order to receive the c/i-indications, it is necessary to use the IPC mechanismus, which is described in this manual. Further information regarding the usage of the C/I monitor is given also in ISDN Application manual.*

4.24.1.4. B CHANNELS RELATED COMMANDS

ENABLE_BCH

(portid / B-chid --)

turns on the selected B channel, so that transmission and receive of data is possible. The portid has to be PORT_BRIDA, PORT_BRIDB or PORT_BRID due to the interface mode NT, TE and mode is one of:

B1_CHANNEL ~ enables the B1 channel transceiver,

B2_CHANNEL ~ enables the B2 channel transceiver.

Note: *If no B Channel Application is running on the FECF's, the B Channels should be disabled to prevent noise in the B Channels.*

DISABLE_BCH

(portid / B-chid --)

turns off the selected B channel, so that no data exchange is possible. The portid has to be PORT_BRIDA, PORT_BRIDB or PORT_BRID due to the Interface mode NT, TE or MONITOR and B-chid is one of:

B1_CHANNEL ~ disables the B1 channel transceiver,

B2_CHANNEL ~ disables the B2 channel transceiver.

BCHEXT**(portid / mode --)**

routes B channels to the BNC jack on the connector module, where portid has to be PORT_BRIDA, PORT_BRIDB or PORT_BRID due to the interface mode NT, TE or MONITOR and mode is one of:

- OFF** – no B channel is routed to the BNC jack on the S-bus connector module,
- B1_CHANNEL** – the B1 channel is routed to the BNC jack on the S-bus connector module,
- B2_CHANNEL** – the B2 channel is routed to the BNC jack on the S-bus connector module.

default: OFF

BCH_SWITCH**(portid / mode --)**

assigns the B1 channel and B2 channel across, where portid has to be PORT_BRIDA or PORT_BRIDB due to the interface mode NT or TE and mode is one of:

- OFF** – assigns B1 channel to PORT 1 and B2 channel to PORT1,
- ON** – assigns B1 channel to PORT2 and B2 channel to PORT1.

default: OFF

BCH_VALID**(portid / B-chid -- / -- state)**

returns the state of the selected B channel, i.e. idle or active, where portid has to be PORT_BRIDA or PORT_BRIDB due to the interface mode and B-chid is one of:

- B1_CHANNEL** – returns the state of B1 channel,
- B2_CHANNEL** – returns the state of B2 channel.

Note: This function has not yet been implemented.

LOOPBACK**(portid / mode --)**

mirrors either B1 channel, B2 channel or both, where portid has to be PORT_BRIDA or PORT_BRIDB due to the interface mode NT or TE, and mode is one of:

- LB_DISABLED** – disables loopback for both B channels,
- LB_ALTER** – loops B1 channel back to B2 and vice versa,
- LB_BOTH** – loops B1 channel back on B1 channel and B2 channel back on B2 channel.

default: LB_DISABLED

Note: This command can't be used in monitor mode.

SWITCH_IN2OUT**(portid / incomming data stream / outgoing data stream --)**

connects an incomming data stream to the ASIC 29 with an outgoing data stream from the ASIC 29, where portid has to be PORT_BRIDA or PORT_BRIDB due to the interface mode NT or TE and incomming/outgoing data streams are one of:

Incomming data streams:

- (0) B1_NT – B1 channel on NT side of BAI board,
- (1) B2_NT – B2 channel on NT side of BAI board,
- (2) B1_TE – B1 channel on TE side of BAI board,
- (3) B2_TE – B2 channel on TE side of BAI board,
- (4) USART3_TX – USART 3 Tx line of CPU2 on FECp,
- (5) USART1_TX – USART 1 Tx line of CPU1 on FECp,
- (6) n.c. – no connection,
- (7) n.c. – no connection,
- (8) EX_TX1 – Tx1 line of external connector,
- (9) EX_TX2 – Tx2 line of external connector,
- (10) BD_IN1 – Burst data Input1 from a second Interface (S2M, BAI),
- (11) BD_IN2 – Burst data Input2 from a second Interface (S2M, BAI),

Outgoing data streams:

- (0) B1_NT – B1 channel on NT side of BAI board,
- (1) B2_NT – B2 channel on NT side of BAI board,
- (2) B1_TE – B1 channel on TE side of BAI board,
- (3) B2_TE – B2 channel on TE side of BAI board,
- (4) USART2_RX – USART 2 Rx line of CPU1 on FECp,
- (5) USART4_RX – USART 4 Rx line of CPU2 on FECp,
- (6) USART3_RX – USART 3 Rx line of CPU2 on FECp,
- (7) USART1_RX – USART 2 Rx line of CPU1 on FECp,
- (8) EX_RX1 – Rx1 line of external connector,
- (9) EX_RX2 – Rx2 line of external connector,
- (10) EX_RX3 – Rx3 line of external connector,
- (11) EX_RX4 – Rx4 line of external connector,
- (12) BD_OUT1 – Burst data Output1 to a second interface (S2M, BAI),
- (13) BD_OUT2 – Burst data Output2 to a second interface (S2M, BAI).

Note: *For a correct understanding of the functions and meaningful connection modes of the ASIC 29, please refer to the Hardware manual for BAI board page 68. The command SWITCH_IN2OUT allows various connections between input and output data streams. The numbers given in brackets above show the input and output parameters.*

Example: PORT_BRIDA 8 0 SWITCH_IN2OUT
 through this example it is possible to transmit data in B1 channel in NT mode, supposed the data source a 64 KBit line.

4.24.1.5. BAI FORTH CONSTANTS

PORT_BRID	(-- value)	BAI Port A (NT) and Port B (TE),
PORT_BRIDA	(-- value)	BAI Port A (NT),
PORT_BRIDB	(-- value)	BAI Port B (TE),
NT	(-- value)	NETWORK,
TE	(-- value)	TERMINAL EQUIPMENT,
P2P	(-- value)	POINT TO POINT OPERATION MODE,
SHORT_BUS	(-- value)	SHORT BUS OPERATION MODE,
EXTENDED_BUS	(-- value)	EXTENDED BUS OPERATION MODE,
B1_CHANNEL	(-- value)	B1 CHANNEL ID,
B2_CHANNEL	(-- value)	B2 CHANNEL ID,
LPRIO	(-- value)	LOW PRIORITY ACCESS,
HPRIO	(-- value)	HIGH PRIORITY ACCESS,
LB_DISABLED	(-- value)	LOOPBACK DISABLED,
LB_BOTH	(-- value)	LOOPBACK ON BOTH CHANNELS,
LB_ALTER	(-- value)	LOOPBACK ALTERNATIVE,
ACTIVATION_IND	(-- value)	ACTIVATION INDICATION OF LAYER 1,
DEACTIVATION_IND	(-- value)	DEACTIVATION INDICATION OF LAYER 1.

4.24.2. CONFIGURATION OF THE PRIMARY RATE INTERFACE

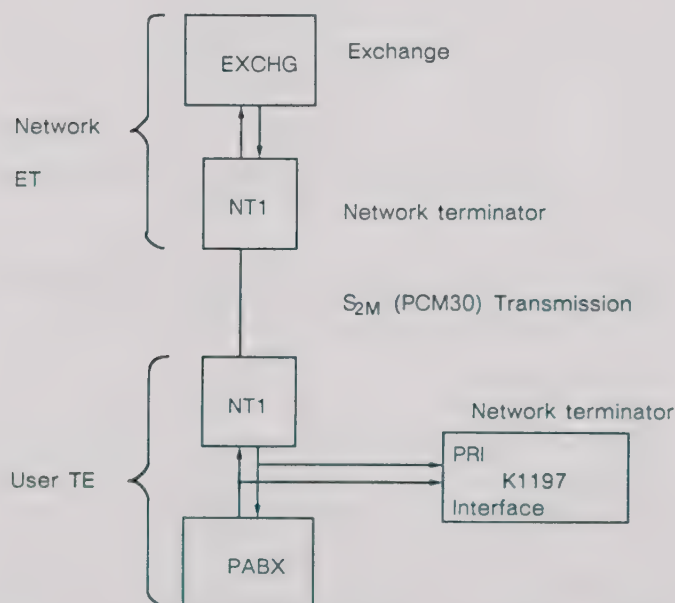
4.24.2.1. INTRODUCTION

Due to the dual Interface concept of K1197, it is possible to install two PRI Interfaces, which are each assigned to only one FECP.

In order to access these Interfaces, the FORTH words PRI1 and PRI2 are introduced as portid's for the first and second PRI Interface. These are further divided in subports PRI1PTA for PRI1 port A, PRI1PTB for PRI1 port B, PRI2PTA and PRI2PTB respectively. For use with the following PRI configuration commands, the argument `pri_pid` should be one of the port ID's listed below:

PRI_PID :	PRI1PTA	(pri 1 port A)
	PRI1PTB	(pri 1 port B)
	PRI1	(pri 1 port A and port B)
	PRI2PTA	(pri 2 port A)
	PRI2PTB	(pri 2 port B)
	PRI2	(pri 2 port A and port B)

Typical measurment (Monitor-State) (Primary Rate Interface PRI)



The PRI Interface is described in hardware specification. There are two variants

- S2M (CEPT-version, PCM30)
- T1 (USA-version, D4, ESF)

4.24.2.2. GLOBAL CONFIGURATION ROUTINES

PRI_I/F_MODE (pri_pid/mode --)

defines the Primary rate network configuration.

In the monitor mode data passes between an active NETWORK and USER and is not disturbed by the tester. In the bidirectional drop and insert mode each port handles one direction of the primary rate link from a NETWORK to a USER. In the emulation mode each PRI port can emulate a USER or NETWORK and are independent of each other. Therefore four different emulation modes are defined to configure each port as NETWORK or USER endpoint:

MODE	PORT_A CONFIGURATION	PORT_B CONFIGURATION
PRI_MON	MONITOR (T-TAP, HIGHIMPEDANCE)	MONITOR (T-TAP, HIGHIMPEDANCE)
DROP_AND_INSERT	DROP_AND_INSERT	DROP_AND_INSERT
UU_EMULATION	USER EMULATION	USER EMULATION
UN_EMULATION	USER EMULATION	NETWORK EMULATION
NU_EMULATION	NETWORK EMULATION	USER EMULATION
NN_EMULATION	NETWORK EMULATION	NETWORK EMULATION

default: PRI_MON

Example: PRI_MON PRI_I/F_MODE
places the PRI_Interface in the T-TAP high-impedance
MONITOR-Mode

GET_PRI_I/F_MODE (pri_pid -- mode)

returns the PRI interface mode. One of the following modes will be placed on the computation stack.

PRI_MON
DROP_AND_INSERT
UU_EMULATION
UN_EMULATION
NU_EMULATION
NN_EMULATION

PRI_FRAMING (pri_pid/value --)

configures the PRI for the specified primary rate framing format.

The impedance is set to high impedance if the PRI is configured for the monitor mode. Additionally it is possible to change the impedance value by using the following FORTH command after the framing format is selected.

Value : D4_FRAMING (at 100 Ohms)
ESF_FRAMING (at 100 Ohms)
PCM_FRAMING (at 120 Ohms)
CRC4_FRAMING (at 120 Ohms)

default: CRC4_FRAMING

PRI_IMPEDANCE (pri_pid/value --)

sets the primary rate source and termination impedances to the specified value.

Value :	HIGH_IMP	(high impedance inputs)
	OHM75	(75 Ohms)
	OHM120	(120 Ohms)

default: OHM120

PRI_LINE_LENGTH (pri_pid/value --)

this command is only applicable in the T1 framing formats (D4, ESF) and should only be used if the T1 connector module for 1.544 Mbps is plugged. For T1 applications, the PRI_LINE_LENGTH command supports transmit line length selection for two different distances.

Value :	LENGTH0	(0 – 220 ft.)
	LENGTH2	(440 – 655 ft.)

default: LENGTH2

PRI_TX_EXT (pri_pid/value --)

selects whether any 64kbps channel in transceiver A and transceiver B of the PRI may be accessed externally. Therefore the selected channels have to be routed (see PRI_ROUTE command) via port PRIPTA. A specific application would be PCM voice – data testing.

Value :	YES
	NO

default: NO

PRI_IDLE_CHAR (pri_pid/char --)

defines the idle character to be transmitted in those timeslots which are programmed to contain the idle character.

default: 0xFF

PRI_LINE_CODE (pri_pid / value --)

in T1 framing modes (D4 and ESF) B8ZS line encoding / decoding is enabled (ON) or disabled (OFF). If one of the CEPT modes (CRC-4 and PCM-30) is selected HDB3 line encoding / decoding is enabled (ON) or disabled (OFF).

PRI_PID :	PRI1PTA	(pri 1 port A)
	PRI1PTB	(pri 1 port B)
	PRI1	(pri 1 port A and port B)
	PRI2PTA	(pri 2 port A)
	PRI2PTB	(pri 2 port B)
	PRI2	(pri 2 port A and port B)

Value :	ON
	OFF

default: ON

PRI_FRMERR_LOC (pri_pid / value --)

this function is only applicable in the CEPT CRC-4 framing format and is ignored in for other framing formats. It defines the framing error generation in bit 3 of EVEN frames or in bit 2 of ODD frames.

PRI_PID :	PRI1PTA	(pri 1 port A)
	PRI1PTB	(pri 1 port B)
	PRI1	(pri 1 port A and port B)
	PRI2PTA	(pri 2 port A)
	PRI2PTB	(pri 2 port B)
	PRI2	(pri 2 port A and port B)

Value : EVEN
 ODD

default: ODD

4.24.2.3. ROUTINES FOR CHANNEL CONFIGURATION**PRI_IHDLC (pri_pid / direction / channel / value --)**

enables or disables inverted HDLC reception or transmission for the selected port and channel.

PRI_PID :	PRI1PTA	(pri 1 port A)
	PRI1PTB	(pri 1 port B)
	PRI1	(pri 1 port A and port B)
	PRI2PTA	(pri 2 port A)
	PRI2PTB	(pri 2 port B)
	PRI2	(pri 2 port A and port B)

Direction : RECEIVER
 TRANSMITTER

Channel : channel number (0 ...31)

Value : YES
 NO

default: all channels non inverted HDLC (data is passed straight through without inversion)

PRI_CHNL_MODE (pri_pid / channel / mode --)

sets the selected channel to the specified mode.

PRI_PID :	PRI1PTA	(pri 1 port A)
	PRI1PTB	(pri 1 port B)
	PRI1	(pri 1 port A and port B)
	PRI2PTA	(pri 2 port A)
	PRI2PTB	(pri 2 port B)
	PRI2	(pri 2 port A and port B)
Channel :	channel number (0 ... 31)	
Mode :	PRI_NORMAL	(data taken from FECF port (PORT1,PORT2,...))
	PASS	(No multiplex -- data is looped from PRI receiver to the associated PRI transmitter. PRI_IHDLC is ignored)
	IDLE	(transmit the idle character)
	TS0_REG	(the international and national bits are transmitted – normally used in TIMESLOT 0)
	RDISC	(receive channel disconnected)

default: all channels IDLE

4.24.2.4. CONFIGURATION AND MONITORING OF TIMESLOT 0**PRI_TNAT_BITS (pri_pid / nat_bits --)**

the defined National bits are only valid in CEPT PCM-30 and CEPT CRC-4 framing format and when the receiver of the specified port is not in the "Red alarm" condition. These bits are the National bits transmitted in timeslot 0 of odd frames and if the timeslot is in the TS0_REG mode (see PRI_CHNL_MODE).

PRI_PID :	PRI1PTA	(pri 1 port A)
	PRI1PTB	(pri 1 port B)
	PRI1	(pri 1 port A and port B)
	PRI2PTA	(pri 2 port A)
	PRI2PTB	(pri 2 port B)
	PRI2	(pri 2 port A and port B)

NAT_BITS : five National bits

default: 0

Note: The National, International and SI bits are only transmitted if timeslot 0 is in TS0_REG mode (see PRI_CHNL_MODE)

PRI_TINT_BIT (pri_pid / int_bit --)
 defines the transmitted International bit in bit 1 of all TS0, if TS0 is in the TS0_REG mode. This bit is only applicable in the CEPT PCM-30 framing format.

PRI_PID :	PRI1PTA	(pri 1 port A)
	PRI1PTB	(pri 1 port B)
	PRI1	(pri 1 port A and port B)
	PRI2PTA	(pri 2 port A)
	PRI2PTB	(pri 2 port B)
	PRI2	(pri 2 port A and port B)

INT_BIT : International bit

default: 0

PRI_TSI_BIT (pri_pid / si_ind / value --)
 sets / clears the SI bits to be transmitted in the CEPT CRC-4 framing format. These bits are ignored in all other framing formats.

PRI_PID :	PRI1PTA	(pri 1 port A)
	PRI1PTB	(pri 1 port B)
	PRI1	(pri 1 port A and port B)
	PRI2PTA	(pri 2 port A)
	PRI2PTB	(pri 2 port B)
	PRI2	(pri 2 port A and port B)

SI_IND :	SI1	(bit1 of timeslot 0 in frame 13 of the multiframe)
	SI2	(bit1 of timeslot 0 in frame 15 of the multiframe)

Value : 0,1

default: both bits 0

PRI_RNAT_BITS (pri_pid -- nat_bits)
 extracts the National bits from timeslot 0 of odd frames. These bits are only valid in CEPT PCM-30 and CEPT CRC-4 framing formats and when the receiver of the specified port is not in the "Red alarm" condition.

PRI_PID :	PRI1PTA	(pri 1 port A)
	PRI1PTB	(pri 1 port B)
	PRI1	(pri 1 port A and port B)
	PRI2PTA	(pri 2 port A)
	PRI2PTB	(pri 2 port B)
	PRI2	(pri 2 port A and port B)

NAT_BITS : National bits

PRI_RSI_BIT (*pri_pid* / *si_ind* / -- *si_bit*)

extracts the specified SI bit from bit 1 of TS0 in frame 15 and bit 1 TS0 in frame 13 respectively. These bits are only valid in the CEPT CRC-4 framing format and when the receiver of the PRI port is not in the "Red alarm" condition.

PRI_PID : PRI1PTA (pri 1 port A)
 PRI1PTB (pri 1 port B)
 PRI1 (pri 1 port A and port B)
 PRI2PTA (pri 2 port A)
 PRI2PTB (pri 2 port B)
 PRI2 (pri 2 port A and port B)

SI_IND : SI1 (bit 1 of TS0 in frame 13 of the multiframe)
 SI2 (bit 1 of TS0 in frame 15 of the multiframe)

SI_BIT : value of the received SI_BIT

4.24.2.5. PRI CHANNEL ROUTING**PRI_ROUTE** (*channel* / *channel-type* / *pri_pid* / *port_d* --)

routes the channel or list of channels to the FECp port (PORT1,PORT2, PRIPTe...) via the selected PRI port.

CHANNEL : channel number (0 ... 31)

CHANNEL_TYPE: B_CHANNEL (single channel number)
 D_CHANNEL (single channel number)

PRI_PID : PRI1PTA (pri 1 port A)
 PRI1PTB (pri 1 port B)
 PRI1 (pri 1 port A and port B)
 PRI2PTA (pri 2 port A)
 PRI2PTB (pri 2 port B)
 PRI2 (pri 2 port A and port B)

PORT_ID : PORT1_A
 PORT1_B
 PORT1
 .
 .
 .
 PORT4_A
 PORT4_B
 PORT4
 PRIPTe

Example: PRI1PTA RX_OFF
 16 D_CHANNEL PRI1PTA PRIPTe PRI_ROUTE
 PRI1PTA CONFIG
 PRI1PTA RX_ON

routes the D_CHANNEL in timeslot 16 of PRI1-PORTA to external port.

4.24.2.6. PRI ERROR / ALARM GENERATION AND REPORTING ROUTINES

PRI_ALARM_GEN (pri_pid / alarm_type / value --)

switches the alarm generation of ALARM_TYPE for the specified PRI port ON or OFF.

PRI_PID :	PRI1PTA	(pri 1 port A)
	PRI1PTB	(pri 1 port B)
	PRI1	(pri 1 port A and port B)
	PRI2PTA	(pri 2 port A)
	PRI2PTB	(pri 2 port B)
	PRI2	(pri 2 port A and port B)

ALARM_TYPE : PRI_YEL_ALARM (Yellow alarm)
PRI_BLUE_ALARM (Blue alarm)

VALUE : ON
OFF

default: OFF

PRI_ERR_GEN (pri_pid / err_rate / tot_rate / err_type --)

configures the selected PRI port to generate errors. The errors are generated in such a way that ERR_RATE errors will occur and, then (TOT_RATE - ERR_RATE) good units will be sent. If the number of errors generated reaches the threshold value ERR_RATE a PRI_ERR_TYPE indication to the calling FORTH machine is sent. Note that PRI_BPVL refers to bits and PRI_CRCR refers to complete CRC's.

PRI_PID :	PRI1PTA	(pri 1 port A)
	PRI1PTB	(pri 1 port B)
	PRI1	(pri 1 port A and port B)
	PRI2PTA	(pri 2 port A)
	PRI2PTB	(pri 2 port B)
	PRI2	(pri 2 port A and port B)

ERR_RATE : number of errors which will occur (ERR_RATE <= TOT_RATE)

TOT_RATE : whole number of units which will be sent

ERR_TYPE :	PRI_BPVL	(PRI Bi Polar violation)
	PRI_CRCR	(CRC error, ESF and CEPT CRC4)
	PRI_FERR	(Frame error, D4, ESF and CEPT CRC4 only)
	PRI_MERR	(Multiframe error, D4 and CEPT CRC4 only)

default: Generation disabled (ERR_RATE = 0 for all ERR_TYPES)

PRI_ERR_STA (pri_pid -- num_of_bpvl/ num_of_crcr / num_of_ferr / num_of_merr)
reports a statistic of total errors detected by the specified PRI port. The number of occurred errors are pushed on the stack in the described order.

PRI_PID :	PRI1PTA	(pri 1 port A)
	PRI1PTB	(pri 1 port B)
	PRI1	(pri 1 port A and port B)
	PRI2PTA	(pri 2 port A)
	PRI2PTB	(pri 2 port B)
	PRI2	(pri 2 port A and port B)

NUM_OF_BPVL : number of BiPolar violations (number of bits)

NUM_OF_CRCR : number of CRC errors (complete CRC)

NUM_OF_FERR : number of Frame errors (number of frames)

NUM_OF_MERR : number of multiframe errors (PCM 30 loss of multiframe alignment)

PRI_ERR_IND (pri_pid / err_type / interval / limit --)
enables an error report after INTERVAL * 100 multiframes received. If no errors are detected no error report is done. The error report contains the number of errors of ERR_TYPE, the report time and, if CEPT special errors PRI_CERR are selected, a byte where the CEPT CRC4 special bits are:

SYNCA : When set (1) this bit indicates that PRI receiver A is able to recognize less than 2 multiframes in 8 msec. This bit may be used as a condition for restarting PRI receiver A by " PRIPTA RX_OFF" and then " PRIPTA RX_ON"

SYNCB : As above for PRI port B

SYNIA : When set (1) this bit indicates that PRI receiver A has not been able to recognize a multiframes alignment signal for greater than or equal to 110 msec.

SYNIB : As above for PRI port B

FH6A : When set (1) this bit indicates that a CRC-4 error rate greater than or equal to 10^{-6} (approx 2 CRC-4 errors per second) is being detected by PRI receiver A.

FH6B : As above for PRI port B

FIMBERR : When set (1) this bit indicates that the PRI receiver is detecting greater than or equal to 915 CRC-4 errors per second.

RMVELA : When set (1) this bit indicates the multiframe "Yellow alarm" condition. This bit is only applicable in the CEPT PCM-30 framing format and when receiver A is not in the "Red alarm" condition.

RMVELB : As above for PRI port B.

PRI_PID :

PRI1PTA	(pri 1 port A)
PRI1PTB	(pri 1 port B)
PRI1	(pri 1 port A and port B)
PRI2PTA	(pri 2 port A)
PRI2PTB	(pri 2 port B)
PRI2	(pri 2 port A and port B)

ERR_TYPE :

PRI_BPVL	(PRI Bi Polar violation)
PRI_CRCR	(CRC error, ESF and CEPT CRC4 only)
PRI_FERR	(Frame error)
PRI_MERR	(Multiframe error, PCM-30 only)
PRI_CERR	(CEPT special error reporting, CRC-4 and PCM-30 only)

INTERVAL : Number of frames in hundreds which are received between two reports

LIMIT : Number of errors

default: no error monitoring

If the number of errors is greater than the specified limit an additional EVENT is sent and the error reporting is stopped. An interval of zero allows error counting until the limit is reached. A limit of zero causes the operating system to repeat the measurement.

4.24.2.7. PRI INDICATION BUFFER EXTRACTION

EXTRACT_PRI (-- address)

extracts the address of the PRI indication buffer from the IPC buffer and places it on the stack. This command normally precedes the UNPACK_PRI command.

UNPACK_PRI (address -- pri_event_type / portid / timestamp_address / blockseqcount / error_count / cept_state)

unpacks a received PRI indication buffer. This command must be preceded by the EXTRACT_PRI command.

Address : address of a PRI indication buffer

PRI_EVENT_TYPE: PRI event type or error type

Portid : contains CPU number and data direction (from user / from network) (0x108, 0x120, 0x200, 0x220)

TIMESTAMP_ADDRESS: address of the timestamp

Blockseqcount : Block sequence count

CEPT_STATE : CEPT special error status
(see 4.24.10)

4.24.2.8. PRI FORTH CONSTANTS

PRI1PTA	(-- value)	PRI PORT A
PRI1PTB	(-- value)	PRI PORT B
PRI1	(-- value)	PORT A AND PORT B
PRI1_PORTA_CBUF	(-- value)	PRI PORT A CONFIGURATION BUFFER
PRI1_PORTB_CBUF	(-- value)	PRI PORT B CONFIGURATION BUFFER
PRI1_PORTA_CHB	(-- value)	PRI PORT A CHANNEL RAM TABLE
PRI1_PORTB_CHB	(-- value)	PRI PORT B CHANNEL RAM TABLE
PRI2PTA	(-- value)	PRI PORT A
PRI2PTB	(-- value)	PRI PORT B
PRI2	(-- value)	PORT A AND PORT B
PRI2_PORTA_CBUF	(-- value)	PRI PORT A CONFIGURATION BUFFER
PRI2_PORTB_CBUF	(-- value)	PRI PORT B CONFIGURATION BUFFER
PRI2_PORTA_CHB	(-- value)	PRI PORT A CHANNEL RAM TABLE
PRI2_PORTB_CHB	(-- value)	PRI PORT B CHANNEL RAM TABLE
PRIpte	(-- value)	PRI EXTERNAL ACCESS PORT
PRI_MON	(-- value)	PRI MONITOR MODE
DROP_AND_INSERT	(-- value)	PRI DROP AND INSERT MODE
UU_EMULATION	(-- value)	PRI USER/USER EMULATION
UN_EMULATION	(-- value)	PRI USER/NETWORK EMULATION
NU_EMULATION	(-- value)	PRI NETWORK/USER EMULATION
NN_EMULATION	(-- value)	PRI NETWORK/NETWORK EMULATION
D4_FRAMING	(-- value)	PRI D4 FRAMING
ESF_FRAMING	(-- value)	PRI ESF FRAMING
PCM_FRAMING	(-- value)	PRI PCM FRAMING
CRC4_FRAMING	(-- value)	PRI CRC4 FRAMING

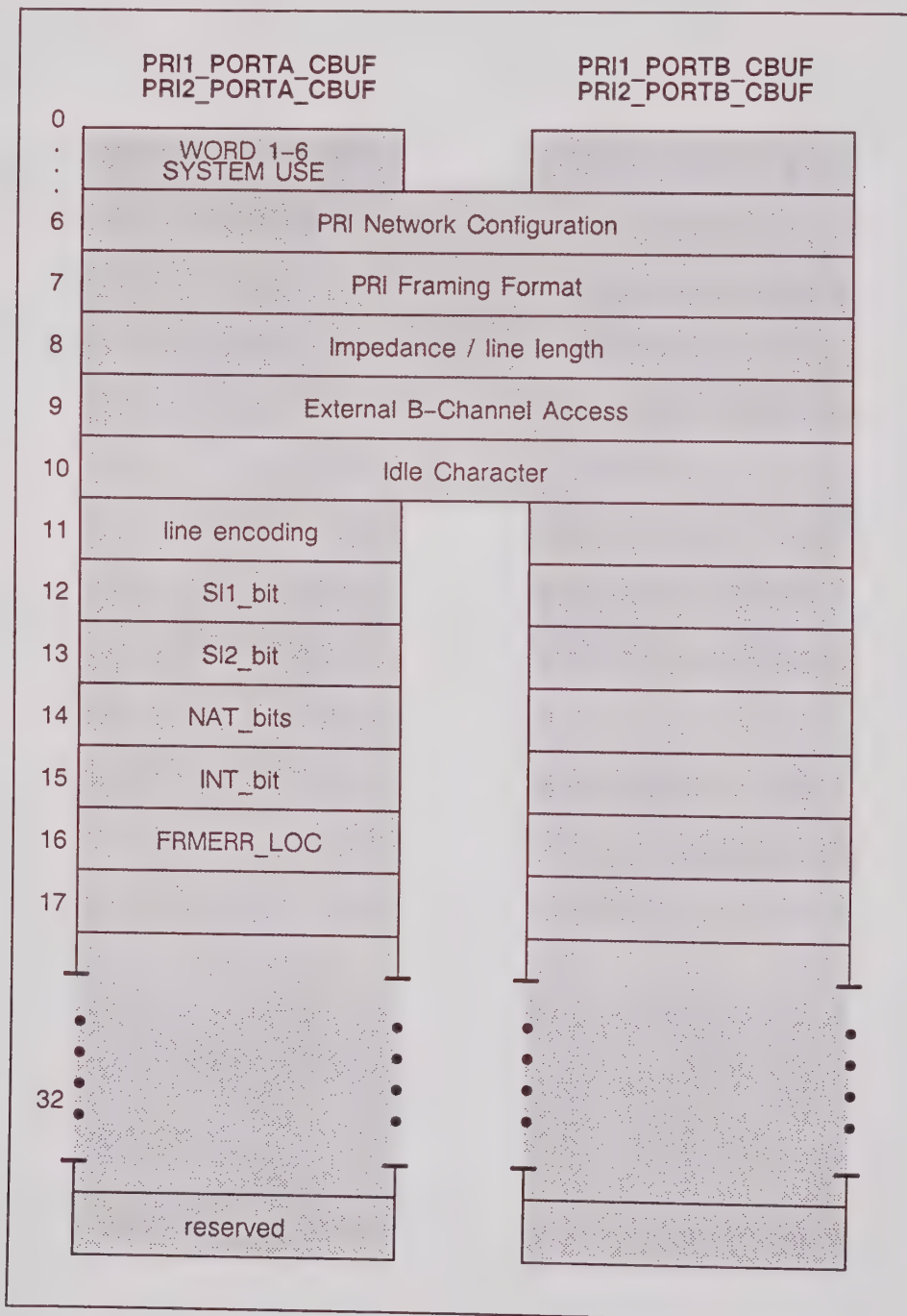
HIGH_IMP	(-- value)	PRI HIGH IMPEDANCE
OHM75	(-- value)	PRI 75 OHM IMPEDANCE
OHM120	(-- value)	PRI 120 OHM IMPEDANCE
EVEN	(-- value)	PRI EVEN LOCATION
ODD	(-- value)	PRI ODD LOCATION
LENGTH0	(-- value)	PRI LINE LENGTH 0-220 FEET
LENGTH2	(-- value)	PRI LINE LENGTH 440-655 FEET
RECEIVER	(-- value)	PRI RECEIVER DIRECTION
TRANSMITTER	(-- value)	PRI TRANSMITTER DIRECTION
PRI_NORMAL	(-- value)	PRI CHANNEL MODE NORMAL
PASS	(-- value)	PRI CHANNEL MODE PASS THROUGH
IDLE	(-- value)	PRI CHANNEL MODE IDLE
RDISC	(-- value)	RECEIVE CHANNEL DISCONNECTED
TS0_REG	(-- value)	PRI TS0 REGISTER
SI1	(-- value)	PRI SI1 BIT
SI2	(-- value)	PRI SI2 BIT
B_CHANNEL	(-- value)	B-CHANNEL
D_CHANNEL	(-- value)	D-CHANNEL
H0_CHANNEL	(-- value)	H0-CHANNEL
H11_CHANNEL	(-- value)	H11-CHANNEL
H12_CHANNEL	(-- value)	H12-CHANNEL
PRI_YEL_ALARM	(-- value)	PRI YELLOW ALARM
PRI_RED_ALARM	(-- value)	PRI RED ALARM
PRI_BLUE_ALARM	(-- value)	PRI BLUE ALARM
PRI_LSL	(-- value)	PRI LOST OF SIGNAL LEVEL
PRI_PLLNL	(-- value)	PRI PHYSICAL LINK NOT LOCKED
PRI_B2NORM	(-- value)	PRI BACK TO NORMAL
PRI_BPVL	(-- value)	PRI BIPOLAR VIOLATION
PRI_CRCR	(-- value)	PRI CRC ERROR
PRI_FERR	(-- value)	PRI FRAME ERROR
PRI_MERR	(-- value)	PRI MULTIFRAME ERROR
PRI_CERR	(-- value)	PRI CEPT SPECIAL ERROR
PRI_EOV	(-- value)	PRI ERROR OVERFLOW
PRI_IND	(-- value)	PRI INDICATION
PRI_LMA	(-- value)	PRI LOST OF MULTIFRAME ALIGNMENT
GEN_BPVL	(-- value)	PRI BIPOLAR VIOLATIONS GENERATED
GEN_CRCR	(-- value)	PRI CRC ERRORS GENERATED
GEN_FERR	(-- value)	PRI FRAME ERRORS GENERATED
GEN_MERR	(-- value)	PRI MULTIFRAME ERRORS GENERATED
TE	(-- value)	TERMINAL EQUIPMENT
NT	(-- value)	NETWORK

4.24.2.9. FURTHER COMMANDS

The commands described below are part of the K1197 FORTH ENHANCEMENTS and may also be used to control or configure the Primary Rate Interface.

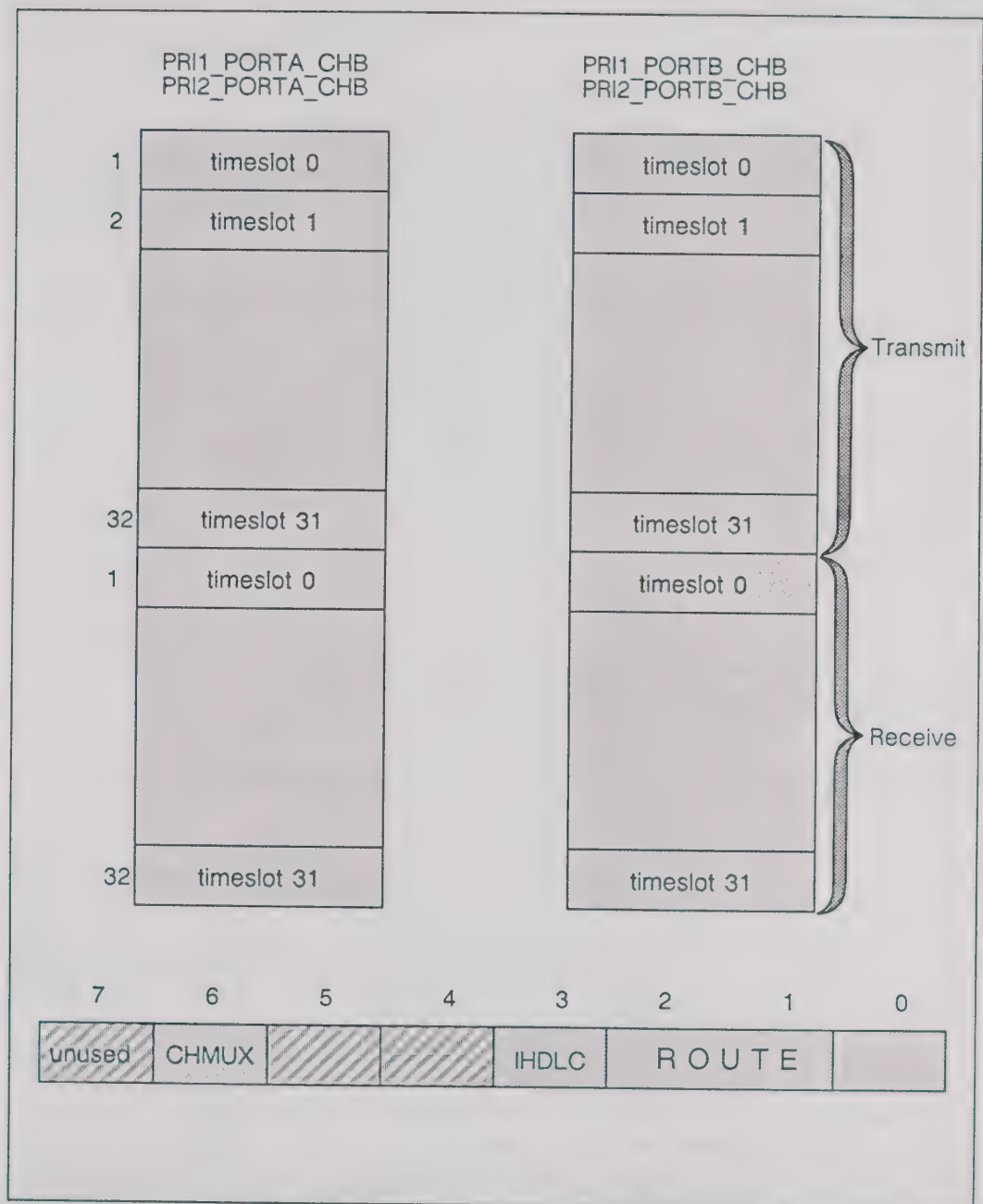
.IBUF	(conf_buf --) displays the 32 word configuration buffer values. The PRI configuration buffers PRI1_PORTA_CBUF, PRI1_PORTB_CBUF, PRI1_PORTA_CHB, PRI1_PORTB_CHB, PRI2_PORTA_CBUF, PRI2_PORTB_CBUF, PRI2_PORTA_ CHB, PRI2_PORTB_CHB.
CONFIG	(port --) performs PRI configuration for the selected PRI port (PRI1PTA, PRI1PTB, PRI1, PRI2PTA, PRI2PTB, PRI2).
RX_ON	(port --) turns receiver on
RX_OFF	(port --) turns receiver off
OPEN_RX	(port / value --) routes the Layer 1 Indications to the calling function. The value may be ON or OFF and is not checked.

4.24.2.10. STRUCTURE OF THE PRI CONFIGURATION BUFFERS



CHMUX:

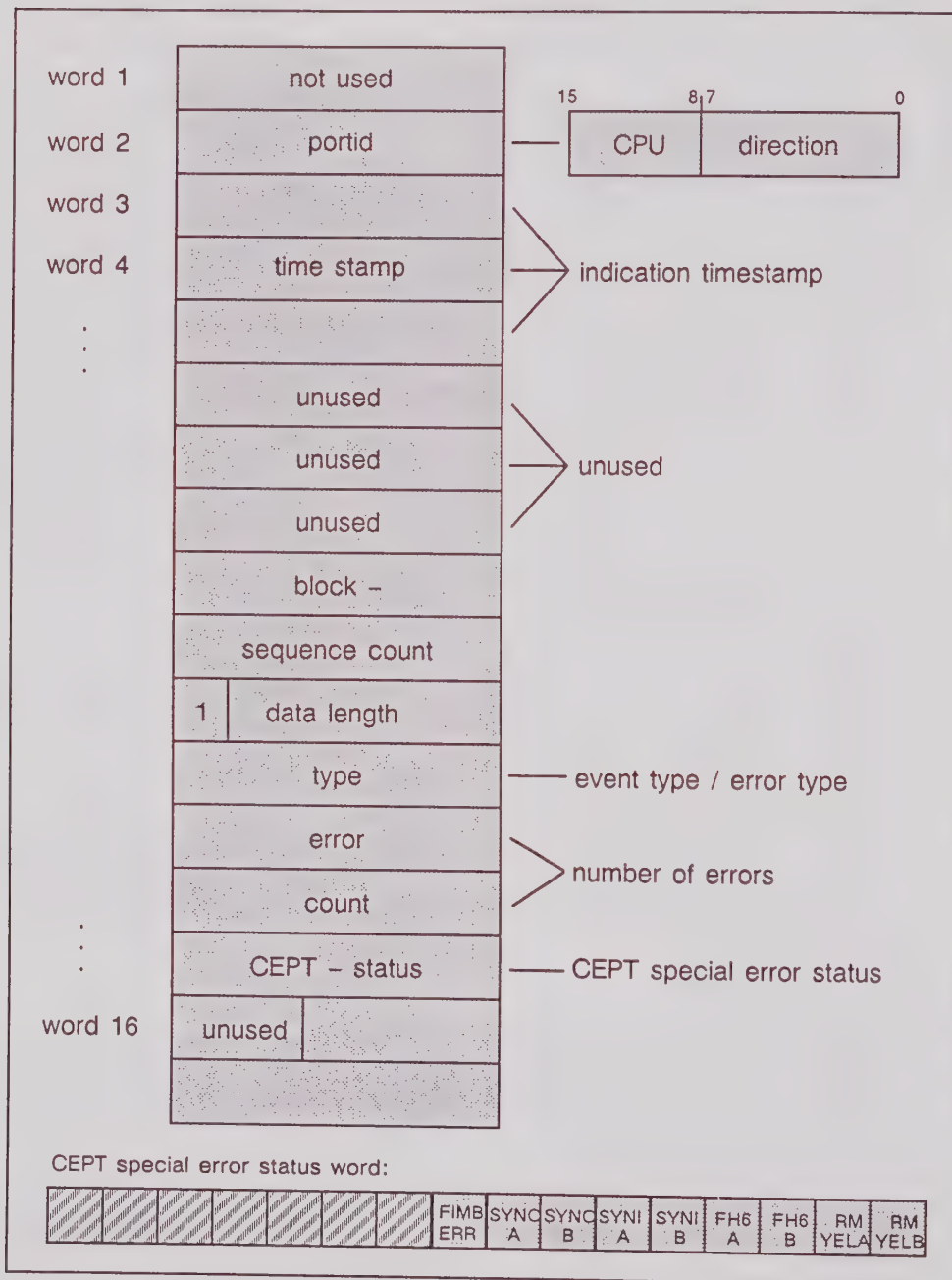
When set (1) this bit indicates that the corresponding timeslot is routed to one of the FECF ports.



IHDLC:
ROUTE:

When set (1) this bit indicates the transmit or receive data stream will be inverted. These bits define the multiplexing / demultiplexing of data streams into the primary rate interface. For detailed description refer to the ISDN PRI INTERFACE FUNCTIONAL SPECIFICATION.

4.24.2.11. STRUCTURE OF THE PRI INDICATION BUFFER



The bits in the CEPT special error status word are described within the scope of the PRI_ERR_IND command in section 4.24.2.5.

4.25. DATA ORIENTED

SEND_DATA (port/buffer/attr --)

Causes a block of data to be sent to the send/receive function.

Where port is the port identification. Buffer is the address of the string to be transmitted. Attribute is one of the following parameters.

SPC	(-- value)	- send steady space
MRK	(-- value)	- send steady mark
CRC	(-- value)	- CRC is to be calculated and transmitted
CRC_ERR	(-- value)	- CRC is to be incorrectly calculated, then transmitted
ABT	(-- value)	- the transmitter is to start transmitting the frame, then aborting part way through
REPT	(-- value)	- the string (or character) contained in BUF will be transmitted continuously.
PAR_ERR	(-- n)	- returns the attribute use by SEND_DATA to generate parity error in ASYNC and COP protocols. The attribute will not have any effect if the parity is not used in the protocol.

SEND_DATA_WAIT (port/buffer/attr --)

Causes a block of data to be sent to the send/receive function. The FORTH machine will not proceed until the data is transmitted. See also SEND_DATA .

Where port is the port identification. Buffer is the address of the string to be transmitted. Attr. is a parameter as described in SEND_DATA .

DATA BUFFER STRUCTURE

WORDS 1-2 : Address where the end of frame timestamp is to be written
 WORD 3 : Byte count
 BYTE 7-N : Data

RECEIVE

Receivers are enabled using the RX_ON command. At this point the receivers are active and are ready to collect data and interface lead status information. The destination function for the data indications is specified using the OPEN_RX command. Only one destination will be active at any one time, and the last OPEN_RX command will override. An indication is returned by the send/receive function to signify the following:

- that a beginning of frame has been received
- that an end of frame has been received
- that an interface lead has changed.

The front end receives the indications and preprocesses the information. After preprocessing, an entry is made in the event queue if the entry for the event has been enabled. Received characters are collected in a buffer area referred to as chunkyfifo. Due to its circular nature, wraparound will take place.

RX_ON (port --)
Turns receiver on, i.e. enables the usart.

RX_OFF (port --)
Turns receiver off, i.e. disables the usart.

I/F_ENABLE (port/Lead_id --)
Enables the indication (via event queue) of interface leads.

Where port may be **PORT1** or **PORT2** and LEAD DESIGNATION may be any combination of leads from the list in Section \$R\$, Interface Lead Control and Checking.

OPEN_RX (port/beg_of_frame_indications? --)
Opens the receive data channel between the receiver and the calling function.

Where beg. of file specifies whether or not beginning of frame indications are to be reported. Beg. of file may be:

(0) OFF – no beginning of frame indications

(1) ON – beginning of frame indications

The default setting is beginning of frame indication OFF.

DATA INDICATION BUFFER STRUCTURE

BEGINNING OF FRAME

WORDS 1-6	: System Use
WORDS 7-9	: Timestamp
WORDS 10-32	: Reserved

INTERFACE LEAD TRANSITION

WORDS 1-6	: System Use
WORD 7	: Port ID (see port definitions for receivers)
WORDS 8-10	: Timestamp
WORDS 11-15	: Reserved
WORDS 16	: 0x8008 (count=4, msb=1 for interface lead ID)
WORDS 17-18	: Interface lead transition status (1=change)
WORDS 19-20	: Interface lead status
WORDS 21-30	: Reserved

Note: For capture buffer and disk recording, only words 7-20 are stored and retrieved.

RECEIVE DATA BUFFER STRUCTURE

WORD 1	: Port ID (see port definitions for receivers)
WORDS 2-4	: Timestamp for beginning of frame
WORDS 5-7	: Timestamp for end of frame
WORDS 8-9	: Block sequence count
WORD 10	: Msb=0 for data ID, byte count
WORD 11	: + data block

4.26. STATE MACHINE RELATED

FUNCTION DEFINITIONS

Multiple FORTH machines are implemented as functions and communicate with each other and RTE modules via standard IPC mechanisms.

4.26.1. EVENT QUEUE MANAGEMENT

Entries for receive data indications, timeout indications, etc, are made in the event queue (FIFO structure) by the front ends when enabled. Control over when entries are to be made is accomplished with the **EVENT_ON** and **EVENT_OFF** commands. When ON, all indications and commands received by the front end will be pushed onto the event queue. All event queue entries consist of a long word and may be copied onto the computational stack using the **POP_EVENT** command.

WAIT_EVENT (--)

Forces the program execution to be suspended until a command or indication has been received.

The complete list of events follows:

COMMAND_IND	(-- value)	- generic command
CURSOR_IND	(-- value)	- cursor key
FK_IND	(-- value)	- function key
I/F_IND	(-- value)	- interface lead transition
PRINT_IND	(-- value)	- printer flag
PRINT_Q_EMP	(-- value)	- printer queue empty
RMT_LEADS	(-- value)	- remote lead transition
RMT_Q_EMP	(-- value)	- remote queue empty
RMT_STR_IND	(-- value)	- string entry from remote port
STR_IND	(-- value)	- keyboard string
SYSERR_IND	(-- value)	- system error indication
TIMER_IND	(-- value)	- timer
BOF_IND	(-- value)	- receiver beginning of frame
EOF_IND	(-- value)	- receiver end of frame
DSM_ERR_IND	(-- value)	- disk error indication
SWITCHED	(-- value)	- this FORTH machine has just been switched to
DSMOVRN	(-- value)	- data recording mechanism is overrun by the received data.

EVENT_OFF (--)

The automatic entry of indications in the event queue will be deleted.

EVENT_ON (--)

Front ends will make entries in the event queue for received indications or commands.

POP_EVENT (-- value)

Pops event indication onto computation stack. If there was no event to pop, 0 will be returned on the stack. When an event is popped, the address of the IPC buffer is automatically saved in a variable called **LAST_IPC**. Prior to actually popping the present event the last IPC message buffer is returned to the originating buffer pool, providing it is not 0 (done automatically by the system).

LAST_IPC**(-- addr)**

The **LAST_IPC** variable is loaded with the address of the IPC message buffer when a **POP_EVENT** is done.

EVENT QUEUE ENTRIES

WORD 1 in all of the following descriptions is **top of the event queue**.

4.26.2. KEYBOARD/PRINTER/REMOTE ENTRIES

CURSOR_IND**FK_IND****PRINT_Q_EMP****RMT_LEADS**

The lead status are in Bits 0 to 3.

bit 0: **RMT_CS** - clear to send

bit 1: unused

bit 2: unused

bit 3: **RMT_CD** - carrier detect

The lead changes are in Bits 4 to 7.

bit 4: **RMT_CS** - clear to send

bit 5: unused

bit 6: unused

bit 7: **RMT_CD** - carrier detect

RMT_Q_EMP**RMT_STR_IND**

RMTGETS **(-- addr/count)**

RMTGETS returns the address and length of the string received from the remote port.

RMT_INTERPRET **(--)**

RMT_INTERPRET interprets the string received from the remote port and returns the flag set by **SET_RETURN**.

RMTGETC **(-- character -1)**

RMTGETC returns a character from the user supplied fifo when the remote port is operating in character mode.

STR_IND

TIMER ENTRIES

TIMER_IND

To aid in extracting the timer value from the IPC message buffer, the command **EXTRACT_TIMER** may be entered.

RECEIVE DATA ENTRIES

indication

EOF_IND	Receiver end of frame
BOF_IND	Receiver beginning of frame
I/F_IND	Interface lead

To aid in extracting the beginning of frame timestamp, end of frame timestamp, and the address of the data buffer the command **EXTRACT_DATA** may be entered.

ERROR ENTRIES

SYSERR_IND

To aid in extracting the error value from the IPC message buffer, the command **EXTRACT_ERROR** may be entered.

4.26.3. IPC BUFFER EXTRACTION

The variable **LAST_IPC** entry following a successful **POP_EVENT** always contains the address of an IPC message buffer (or 0 if not applicable). Recovering the information from the buffer may appear to be difficult, and to simplify things for the user, a set of commands is available which extracts the useful information and places it on the computational stack.

EXTRACT_KEY	(-- function_or_cursor_key_value) Extracts the key value (function key or cursor key) from the IPC buffer and places it on the stack.
EXTRACT_TIMER	(-- timer_no) Extracts the timer value from the IPC buffer and places it on the stack.
EXTRACT_BOF_TS	(-- beg_of_frame_time_stamp) Extracts the address of the timestamp from the beginning of frame indication buffer (IPC) and places it on the stack. (See GET_TSTAMP_MICRO and GET_TSTAMP_MILLI)
EXTRACT_DATA	(-- data_buffer_addr) Extracts the address of the data buffer from the IPC buffer and places it on the stack. This command normally precedes the UNPACK_DATA command.
EXTRACT_LEADS	(-- addr) Extracts the address of the lead transition buffer and places it on the stack. This command normally precedes the UNPACK_DATA command.

UNPACK_DATA (data_addr -- status/data/BSC/end_timestamp/start_timestamp/port)

Unpacks a received data or interface lead transition buffer. This command must be preceded by the **EXTRACT_DATA** or **EXTRACT_LEADS** commands. The information is placed on the computational stack as follows:

For DATA:

Top of Stack:	1 (DATA INDICATION)
	PORT ID
	START TIMESTAMP ADDRESS
	END TIMESTAMP ADDRESS
	BLOCK SEQUENCE COUNT
	DATA LENGTH
	DATA ADDRESS
Bottom of Stack:	STATUS BYTE

For INTERFACE LEAD TRANSITIONS:

0 (LEAD TRANSITION INDICATION)
PORT ID
TIMESTAMP ADDRESS
LEAD CHANGE MASK
LEAD STATUS

Where BUFFER ADDRESS is the address returned by the **EXTRACT_DATA** command.

RX_CH1 (-- TO_DCE rx id for port 1/TO_DTE rx id for port 1)

RX_CH2 (-- TO_DCE rx id for port 2/TO_DTE rx id for port 2)

RX_CH3 (-- TO_DCE rx id for port 3/TO_DTE rx id for port 3)

RX_CH4 (-- TO_DCE rx id for port 4/TO_DTE rx id for port 4)

Returns the TO_DCE and TO_DTE receive stream ID for the PORT 1, PORT 2, PORT 3, PORT 4, PORT 5 and PORT 6, respectively. These ID's are used to match the PORT ID returns by the **UNPACK_DATA** command.

MYRX_CH (-- TO_DCE rx stream id/TO_DTE rx stream id)

Returns the TO_DCE and TO_DTE receive stream id for the port that corresponds to the FORTH machine. These ID's are used to match the **UNPACK_DATA** command.

EXTRACT_ERROR (-- function_no/error)

Extracts the error associated with a **SYSERR_IND** event. The function no. is the function number of the function that is in error. (See list under **MAIL**) The error will be one of the following:

- | | |
|-----------------|---|
| RXEMAIL | - Receiver can't mail received frame to the connected function. |
| RXE_FIFO | - Receiver's FIFO underrun. |
| RXEBUF | - Receiver runs out of message buffer to send received frame to the connected function. |
| I/FEMAIL | - Receiver can't mail interface lead changes to the connected function. |
| I/FEBUF | - Receiver runs out of message buffer to send interface lead changes to the connected function. |
| FTHEBUF | - FORTH couldn't buy message buffer. |
| FTHEMAIL | - FORTH couldn't mail IPC buffer. |

EXTRACT_DSM_ERR (-- error_value)

Extracts the disk error value which occurs during data recording.

Use **EXTRACT_DSM_ERR RPT_D_ERR** to display the error message on the screen.

EXTRACT_DSMORVN (-- sequence_number/R_x_Stream_id)

Extracts the receiver stream id and the sequence number of the received block when DSM was overrun. This information is passed along with the **DSMORVN** indication and its message buffer address is stored in **LAST_PIC**.

4.26.4. MAILING

The following words are for mailing to other processes.

MAIL

(function_no/buffer --)

Mails a buffer (standard 32 word format) to another function. Where function no. is one of the following list (function numbers are in the ranges):

- 1 – 31 for main cpu
- 33 – 63 for fecp1a
- 65 – 95 for fecp1b
- 97 – 127 for fecp2a
- 29 – 159 for fecp2b
- 161 – 191 for fecp3a
- 193 – 223 for fecp3b

- 1 – PPA1
- 2 – TCF
- 3 – DIO
- 4 – Data Stream Merger
- 5 – Time function
- 6 – Editor
- 8 – FORTH machine
- 33 – PPA2
- 34 – USART3 (PORT2 B)
- 35 – USART4 (PORT1 A)
- 37 – FTH –FORTH machine
- 65 – PPA3
- 66 – USART1 (PORT2 A)
- 67 – USART2 (PORT1 B)
- 69 – FTH–FORTH machine

BUF is the address of the buffer which has been prepared by the FORTH program.

MAIL_WAIT

(function no./buffer --)

Mails a buffer (standard 32 word format) to another function and waits for a response, i.e. this FORTH machine goes to sleep eg. calling the edit function.

Where function no. is one of the list shown under 'MAIL'. Buffer is the address of the buffer which has been prepared.

RE_ROUTE

(function no. --)

Causes the message buffer currently being processed to be rerouted to another function. Where function no. is the function to which the last IPC buffer is to be re-routed.

MES_BUF (-- 0|buffer_addr)

Returns the address of a 32 word message buffer or 0 if no buffer is available in the FORTH buffer pool. It is important to check the return value from MES_BUF because of the 0 returned if there is no more message buffer available in FORTH. If the program ignores the 0, it will overwrite the vector address at location 0 and cause the operating system to crash.

STRUCTURE OF A MESSAGE BUFFER

WORD 1 :	Request/response
WORD 2 :	Originator's function number
WORD 3 :	Argument count
WORD 4-5 :	External buffer address
WORD 6 :	Destination count
WORD 7-32:	Specified by each request/response

MBSTATUS (-- 0|buffer_addr)

MBSTATUS is used to save the status of the last message buffer bought from the system. It returns the address of the last message buffer bought if successful or a 0 if a failure.

DIO (-- value)

Disk I/O function number.

DSM (-- value)

Data stream merger (data recording) function number.

Returns the **DSMOVRN** indication when it overruns by the receive frames. The receiver stream id and the sequence number of the received frame are also sent along with this indication. This information can be extracted by **EXTRACT_DSMOVRN**.

TCF (-- value)

Terminal/console function number. May be used by **MAIL**, **MAIL_WAIT**.

EDF (-- value)

Editor function number.

FTH (-- value)

Function number for this FORTH function.

PPA (-- value)

PPA function number.

USART1 (-- value)

PORT2_A function number.

USART2 (-- value)

PORT1_B function number.

USART3 (-- value)

PORT2_B function number.

USART4 (-- value)

PORT1_A function number.

USART5 (-- value)
 PORT4_A function number.

USART6 (-- value)
 PORT3_B function number.

USART7 (-- value)
 PORT3_A function number.

USART8 (-- value)
 PORT4_B function number.

4.26.5. KEY - CONSTANTS

FUNCTION KEYS

UF1	(-- value)	Function key f1 value.
UF2	(-- value)	Function key f2 value.
UF3	(-- value)	Function key f3 value.
UF4	(-- value)	Function key f4 value.
UF5	(-- value)	Function key f5 value.
UF6	(-- value)	Function key f6 value.
UF7	(-- value)	Function key f7 value.
UF8	(-- value)	Function key f8 value.
SF1	(-- value)	Function key SHIFT f1 value.
SF2	(-- value)	Function key SHIFT f2 value.
SF3	(-- value)	Function key SHIFT f3 value.
SF4	(-- value)	Function key SHIFT f4 value.
SF5	(-- value)	Function key SHIFT f5 value.
SF6	(-- value)	Function key SHIFT f6 value.
SF7	(-- value)	Function key SHIFT f7 value.
SF8	(-- value)	Function key SHIFT f8 value.
CF1	(-- value)	Function key CTRL f1 value.
CF2	(-- value)	Function key CTRL f2 value.
CF3	(-- value)	Function key CTRL f3 value.
CF4	(-- value)	Function key CTRL f4 value.
CF5	(-- value)	Function key CTRL f5 value.
CF6	(-- value)	Function key CTRL f6 value.
CF7	(-- value)	Function key CTRL f7 value.
CF8	(-- value)	Function key CTRL f8 value.

CURSOR KEYS

CHOME	(-- value)	HOME cursor key value.
CUP	(-- value)	UP arrow cursor key value.
CDOWN	(-- value)	DOWN arrow cursor key value.
CRIGHT	(-- value)	RIGHT arrow cursor key value.
CLEFT	(-- value)	LEFT arrow cursor key value.

4.27. CODE CONVERSION

The CONV command allows for conversion from one code set to another one.

CONV (*source/count/dest/code* -- *-1|count/0*)
Performs code conversion of the source string and puts the result in the destination. If it encounters a byte that it cannot convert, then the byte is replaced by a '?' in the destination.

Where the codes supported are:**ASCII-TO-EBCDIC**
 EBCDIC-TO-ASCII

If an illegal conversion code type is specified, then -1 is returned. If the conversion is successful, then 0 and the number of bytes converted are returned. If the conversion is partially OK, then the number of unconverted bytes and the number of bytes converted successfully are returned.

EVEN_PAR_BIT (*char* -- *even parity bit of char*)
Returns the value of the even parity bit of the given character.

4.28. CRC CALCULATION

CRC_CALC (*addr/count/crc type* -- *MSB/LSB*)
Calculates the CRC for the string at addr with a length of count. The CRC types are specified as **CRC_16** or **CRC_CCITT**. The CRC values are given as the most significant byte (MSB) and least significant byte (LSB).

LRC8 (*addr/count* -- *LRC8*)
Calculates and returns the LRC8 for the string at addr with a length of count.

LRC_EVEN (*addr/count* -- *even LRC*)
Calculates and returns the even LRC value for the given string.

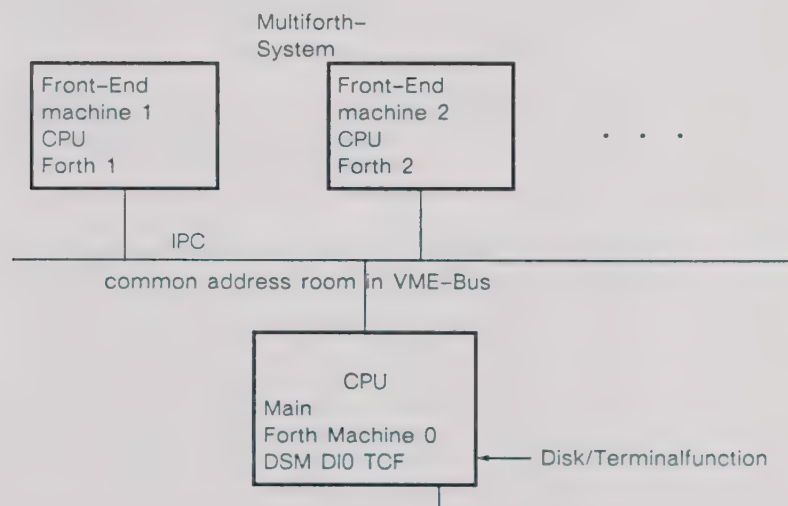
LRC_ODD (*addr/count* -- *odd LRC*)
Calculates and returns the odd LRC value for the given string.

5. K1197 MULTIFORTH ENHANCEMENTS

The advent of the K1197 Operating System requires basic interprocess communication and control mechanisms available to the programmer. The single FORTH machine RTE used the file ALOAD and the word ALOAD as the entry point for automatic bootup execution. The new environment uses the same mechanism for the MAIN or MFTH FORTH machine but uses ALOAD1 and its own ALOAD colon definition for FEF_1, likewise, FEF_2 uses an ALOAD2 file and again another ALOAD colon definition. Each FORTH environment appears exactly the same as the old RTE and runs completely independent of each other. The requirement for interprocess communication comes with the attempted use of a single system wide resource such as the Display, keyboard, or the remote port.

For example, this release allows for only a limited amount of display memory (the same as with the old RTE). In this case, the 'non-visible' windows of an application may have to be sold back to the display manager in order to preserve the integrity of the displayed windows. Interprocess communication would be required to coordinate the buying and selling of this limited resource.

(Communication between processes = Interprocess Communication = IPC).



The terminal (keyboard + screen) will be connected through the MAIN-CPU with the actual communicating process.

5.1. K1197 INTERPROCESS COMMUNICATION WORDS

The following words have been added for control and communication between the environments.

FEF_COLD (FORTH# --)

Causes the specific FORTH machine to perform COLD

FORMAT: FORTH# FEF_COLD

FEF_DO_WORD (" cmds"/FORTH# --)

Executes the commands as if they were typed in on the keyboard on the designated FORTH machine.

FORMAT: " cmds" FORTH# FEF_DO_WORD

FEF_GOFORTH (FORTH# --)

Causes the specific FORTH machine to perform GOFORTH.

FORMAT: FORTH# FEF_GO_FORTH

SWITCH (FORTH# --)

Connects to the designated FORTH machine. The designated FORTH machine will receive the SWITCHED indication if event processing is on.

FORMAT: FORTH# SWITCH

WARNING: The system will lock up if the argument passed to SWITCH does not correspond to the function number of any of the FORTH machines.

PACK_FTH_DATA (d1/d2/.../# of d's -- message_buffer_addr| -1)
 Packs the number of variables into a message buffer and inserts CMD_FLAG value in the buffer which interprets it as an event (indication) by the designated FORTH machine. This is used by inter-FORTH communication. There is a limit of 12 values allowed. PACK_FTH_DATA returns -1 if it cannot get a message buffer. Otherwise, the address of the message buffer is returned.
 FORMAT: d1/d2/.../# of d's PACK_FTH_DATA. In the case where buffer allocation failed the RUN_PRE System function should be executed.

EXTRACT_FTH_DATA (-- d1/d2/.../#_of_d's)
 Returns the values from the message buffer to the stack with the number unpacked at the top. Uses LAST_IPC.
 FORMAT: EXTRACT_FTH_DATA

COMMAND_IND (-- value)
 Returns the value of COMMAND_IND.
 FORMAT: COMMAND_IND

MF1.5 or MF1.6 (--)
 identifies the version of operating system.

MFTH (-- Main_FORTH's_id)
 Returns the ID for the FORTH machine on Main CPU.
 FORMAT: MFTH

FEF_1 (-- FORTH_#1's_id | Current_FORTH's_id)
FEF_2 (-- FORTH_#2's_id | Current_FORTH's_id)
FEF_3 (-- FORTH_#3's_id | Current_FORTH's_id)
FEF_4 (-- FORTH_#4's_id | Current_FORTH's_id)
FEF_5 (-- FORTH_#5's_id | Current_FORTH's_id)
FEF_6 (-- FORTH_#6's_id | Current_FORTH's_id)
FEF_7 (-- FORTH_#7's_id | Current_FORTH's_id)
FEF_8 (-- FORTH_#8's_id | Current_FORTH's_id)
 Returns the ID for the FORTH machine on CPU 1 (Fecp1a), CPU 2 (Fecp1b), CPU 3 (Fecp2a), CPU 4 (Fecp2b), CPU 5 (Fecp3a), CPU 6 (Fecp3b), CPU7 (BAI1), CPU8(BAI2) respectively. If the designated FORTH machine does not exist, then an error message is printed on the FORTH window and ID of the current FORTH is returned on the stack.

Note:

- i) ALOAD1 is the autoload filename for FEF #1. (FEF1a)
- ii) ALOAD2 is the autoload filename for FEF #2. (FEF1b)
- iii) ALOAD3 is the autoload filename for FEF #3. (FEF2a)
- iv) ALOAD4 is the autoload filename for FEF #4. (FEF2b)
- v) ALOAD5 is the autoload filename for FEF #5. (FEF3a)
- vi) ALOAD6 is the autoload filename for FEF #6. (FEF3b)
- v) ALOAD7 is the autoload filename for BAI1
- vi) ALOAD8 is the autoload filename for BAI2.

FEF_7 (-- FORTH_#7's_id | Current_FORTH's_id)

FEF_8 (-- FORTH_#8's_id | Current_FORTH's_id)

(NFORTH) (-- mask)

Returns the mask for the number of FORTH machines that exist in the system. Each bit in the mask represents a FORTH machine. Bit 0 corresponds to the Main FORTH machine, bit 1 corresponds to front end FORTH machine number 1, and bit 2 corresponds to front end FORTH machine number 2 and so on. The FORTH machine exists if the corresponding bit value is 1.

NFORTH (-- number_of_FORTH_machines)

Returns the number of FORTH machines that exist in the system.

6. PROGRAM DEBUG TOOLS

FIND (--)
Used in the form: FIND nnnn

Finds and prints all definitions that contain the dictionary word nnnn.

LOCATE (--)
Used in the form: LOCATE nnnn

Locates the word nnnn and displays the source filename in where the word is located.

DECOMP (--)
Used in the form: DECOMP nnnn

Decomposes and prints the dictionary word nnnn.

	output
0 hz	vertical
1 hz	horizontal

DECOMP.H (--)
Used in the form: DECOMP.H nnnn

Decomposes and prints the dictionary word nnnn with memory address in HEX.

DECOMP_H (--)
Used in the form: DECOMP_H nnnn

Decomposes and prints the dictionary word nnnn horizontally.

DECODE (PFA/d --)
Decomposes and prints the dictionary word designated by PFA up to 'd' dictionary words.

ALTER (--)
Used in the form: ALTER nnnn

Alters the definition of the dictionary word nnnn. Used with **FROM** and **TO**.

FROM (--)
Used in the form: FROM aaaa

Defines the cell aaaa in the dictionary word nnnn to be altered. Used with **ALTER** and **TO**.

-
- TO** (--)
Used in the form: TO bbbb
- Defines the cell aaaa in the dictionary word nnnn to be replaced by bbbb. Used with **ALTER** and **FROM**.
- For Example:** ; ABC 2 1 + . ;
ALTER ABC FROM + TO -
changes the '+' operation in ABC to '-'
- UNALTER** (--)
Used in the form: UNALTER nnnn
- Undefines all the changes to the dictionary word nnnn.
- NULLIFY** (--)
Used in the form: NULLIFY nnnn
- Changes the definition of the dictionary word nnnn to do nothing.
- CHANGE** (-- PFA)
Used in the form: CHANGE nnnn
- Changes the meaning of a word into another word. Used with **INTO**.
- INTO** (PFA --)
Used in the form: CHANGE nnnn INTO ssss
- Replaces a word nnnn with the word ssss.
- For Example:** CHANGE MENU INTO GOFORTH
MENU would now bring you to FORTH instead of the menu when it is executed.
- FILES** (-- filename_specifiers)
Used in the form: MAKE FILES 0 " FILE_2.F" " FILE_1.F" -1 ;
- FILES is a DOER word that is used by LOAD_FILES to specify the files names to be compiled. 0 is used to specify the End of Files, -1 selects DR0 disk drive, and -2 selects DR1 disk drive.
- LOAD_FILES** (filename_specifiers --)
Used in the form: FILES LOAD_FILES
- REDO** (--)
Used in the form: REDO FILENAME
- Performs a re-compilation starting at FILENAME.
- !!** (--)
Enables the cursor to be moved inside the commanded-windows to repeat previously given commands.
- Note: Not applicable for MAIN_CPU.
-

SEARCH (STRING ---)

Displays all Forth_Definitions, which contain 'string' in their name-field.

QUICKY (MODE ---)

where mode is one of:

ON Fast compiler is activated.

OFF Fast compiler is deactivated.

Note: Compiling is done using MASM_mechanism (I.g. useful when compiling Testscenarios).

COME_BACK (--)

If a runtime error occurs (i.e. BUS ERROR), a part of the screen turns to red, and while pressing return the return stack backtrace will appear on the screen.

COME_BACK displays the red screen again. (i.e. for printing).

7. FORTH ERROR MESSAGES

The following error messages have been defined. They may be displayed using the 'MESSAGE' command. eg. MESSAGE.

7.1. FORTH EXECUTION/COMPILATION ERROR MESSAGES

MESSAGE NO.	Description
1	Stack empty
2	Dictionary full
3	Has incorrect address mode
4	Is not unique in the dictionary
7	Computational stack is full
8	Disk error
17	Use in compilation only
18	Use in execution only
19	Conditionals not paired
20	Definition not finished
21	In protected dictionary
22	Use only in compiling/loading
24	Declare vocabulary
30	Use in 'Smart Mode' only
31	No return set up
32	Bad HEX string

7.2. DISK OPERATION ERROR MESSAGES

MESSAGE NO.	Description
40	No disk error
41	Invalid disk command
42	Error in command argument(s) to disk I/O function
43	File does not exist
44	Unrecoverable physical I/O error
45	Nonexistent I/O device
46	Bad file descriptor
47	File access prohibited
48	File already exists
49	Error in disk I/O function
50	File name error
51	End of file
52	No space for file system
53	Too many files opened
54	File system busy
55	Floppy disk not loaded
56	No 'RTE' filesystem
57	Disk is write protected
58	Can't copy file to itself
59	Bad source device
60	Bad destination device
61	No match for wildcards
62	Difference in compared files
63	File system is write protected

7.3. K1197 ERROR MESSAGES

MESSAGE NO.	Description
-	Non-existent Test Port
-	Non-existent FORTH Machine

8. TEST MANAGER

Note: The Test Manager and its commands can only be utilized if an application program containing the Test Manager Software is in memory.

PROGRAMMERS GUIDE TO THE STATE MACHINE ROUTINES

The state machine routines are provided to allow the programmer to easily create efficient state machine representations. It is assumed that the readers of this document will be application programmers who will be doing a protocol decode before the state machine is run. Thus, routines that do console I/O manipulations are omitted.

8.1. OVERVIEW

There can only be one state machine in existence at once.

The state machine is implemented using FORTH words whose actions cause compilation to be redirected to a pseudo dictionary with additional control code added to perform the state execution.

The pseudo dictionary is an array in the memory. Thus, there is a limit to the size of the total state machine; 256 states are presently supported.

Any number of states can be defined, subject to the limit described in the previous paragraph. States do not have to be defined in any particular sequence, i.e. state 0, then state 1, state 2, etc. States do not have to be defined at the time they are referenced. However, FORTH words that are used must be defined.

If a reference is made to a state that is not defined, no error is generated until the state machine is run and there is an attempt to actually change to that state.

FORTH words defined by the application or by the system can be used within the state machine. Actually, the best way to think of the state machine words is like another programming language construct, rather like an extended DO CASE statement. In the same way that one must be careful how FORTH words are used in the DO CASE, one must be careful with the state machine.

The following FORTH words are provided to build a state machine.

state_number	STATE{ }STATE
event_comparison	ACTION{ }ACTION
state_number	NEW_STATE
state_number	LOAD_RETURN_STATE
	RETURN_STATE
state_number	STATE_INIT{ }STATE_INIT
sequence_number	SEQ{ }SEQ
sequence_number	RUN_SEQ

```
STATE{      ( state_number -- )
}STATE      ( -- )
```

The **STATE{ }STATE** words bracket the definition of a single state within a state machine. **STATE{** expects a single integer which is the state number. If two states are defined with the same number, the second definition is used with no error messages.

```
ACTION{      ( truth_flag -- )
}ACTION      ( -- )
```

The **ACTION{ }ACTION** words bracket the FORTH code to be executed if the associated event occurs. **ACTION{** expects a truth flag on the stack. If the truth flag is non-zero, the action list inside **ACTION{ }ACTION** is performed. If the truth flag was zero, the next event is checked. The events are defined by FORTH code that returns a flag to be used by the **ACTION{** word. If the present event comparison results in a false condition (0), the next event is compared against the received event. There are high level words provided to perform the event comparisons needed. eg. **FK_CF1**, **TIMER**.

```
NEW_STATE    ( state_number -- )
```

Expects a single integer and performs a change of state. **NEW_STATE** is only legal within an **ACTION{ }ACTION** pair. Another way to change the state is to modify the contents of the variable that contains the state. Thus **1 NSTATE !** does the same thing as **1 NEW_STATE**.

```
RETURN_STATE ( -- )
```

Permits the user to program the equivalent of subroutine calls into the state machine. The procedure is as follows:

1. The action word is generated which will cause an event to be received.
2. Load the state number of the state to which control is to be returned:
X **NEW_STATE**.
3. Load the subroutine state value using: X **NEW_STATE**.
4. When the end of the subroutine has been reached, execute:
RETURN_STATE.

Nesting is not permitted, only one level is supported at this time.

Below is a short example:


```
0 STATE{  FK_CF1
          ACTION{
              START_TEST 1 NEW_STATE
          }ACTION
        }STATE
```

Note that like other FORTH words, state machines can be formatted in any way that suits the programmer.

The following FORTH words are used to initialize, run and stop the state machine:

TCLR	(--) initializes the state machine data structure. Any existing Test Scenarios for the currently active TM will be cleared. The current state is set to 0. It is advisable to start new test scenarios with a TCLR command before defining the new state machine.
TM_RUN	(--) starts the state machine running in state 0. The events in state 0 will be checked and when any of them happen, the defined action will be executed.
TM_STOP	(--) is used to terminate the operation of the Test Manager state machine.

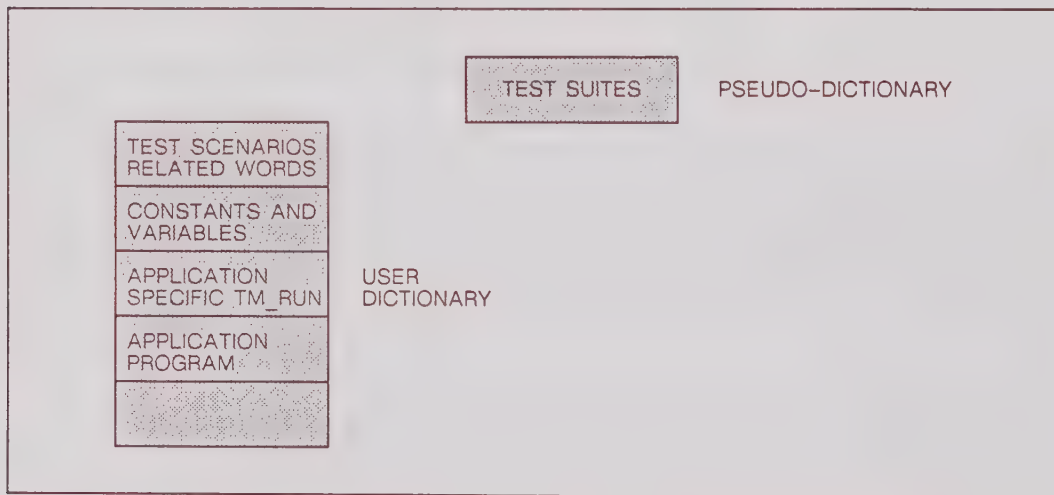
Another way to change the state is to modify the contents of the variable that contains the state. Thus 1 NSTATE ! does the same thing as 1 NEW_STATE.

To load a state machine, simply put the definition in a file and EXEC it. State machines cannot be defined inside a ':' definition.

DEVELOPING TEST SCENARIOS WHICH UTILIZE CAPABILITIES OF EXISTING APPLICATIONS PROGRAMS

Siemens developed application programs, have capabilities which are accessible to the user in various ways. The typical way allows the user to enter commands in the command line and have them interpreted. A more automated approach, requires the user to program a control sequence to reside on top of the application. This approach requires the user to have detailed knowledge of how to design a main control loop, and how to effectively call the various words residing in the application.

An alternate approach allows the user to program test scenarios which are to be executed by the test manager. The applications programs contain some interface words which simplify the programming somewhat, eg. ?RX_FRAME and ?RX in the X.25 reference emulation. The TM_RUN is duplicated in each reference emulation and is custom tailored to the application. Any other support words used to initialize test scenarios, or as designed by the user to be called on the test scenario execution must also be compiled into the dictionary. The following diagram describes the organization.



9. DESIGN NOTES

9.1. KEYBOARD MANAGEMENT

Different sections of the keyboard may be owned by various functions, for instance the unshifted function key group could belong to the editor, while the shifted function key group could belong to the controlling application program which called the editor. This capability carries with it the necessity to buy all keyboard sections that are required.

EXAMPLE: An application requires the entire keyboard. The following sequence should be employed:

LOCAL KBD CURSOR OR UF OR SF OR CF OR BUY_KEYS

The dictionary contains constants that uniquely identify the function key and cursor keys as follows:

Unshifted function keys	UF1...UF8
Shifted function keys	SF1...SF8
Control function keys	CF1...CF8
Up arrow cursor key	CUP
Down arrow cursor key	CDOWN
Left arrow cursor key	CLEFT
Right arrow cursor key	CRIGHT
Home cursor key	CHOME

A typical keyboard decode example follows. We assume here that the POP_EVENT logic has recognized that a function key has been entered.

```
EXTRACT_KEY
DOCASE
    CASE UF1  {...}
    CASE UF2  {...}

    CASE SF1  {...}

ENDCASE
```

9.2. HANDLING KEYBOARD INPUT STRINGS UNDER PROGRAM CONTROL:

IMPLEMENTING A COMMAND LINE

Interpretive FORTH operates on each character as it is entered. As soon as event queue entry is woken up, keyboard entry is more tightly controlled by the front ends. Function and cursor keys are immediately passed on to the event queue, however strings are held in a special input buffer in the front ends. Once the keyboard indication has been detected in the event queue, the string can be extracted using QUERY (which transfers the string to the terminal input buffer), or EXPECT (which transfers the string into the user specified buffer).

When an application has been programmed to be run by an infinite control loop, the FORTH interpreter is no longer active. The FORTH interpreter can be instructed by the program to interpret

user input. This can be accomplished by defining a command line. When this line is active, the cursor should be positioned at the beginning of this line and a status flag set. The program will be able to accept strings which are to be interpreted, or used as strings eg. filenames in response to queries. The program differentiates between these two usages by examining the command flag.

```
( KEYBOARD STRING HAS BEEN IDENTIFIED )
```

```
COMMAND-FLAG @          ( GET COMMAND FLAG )
IF  QUERY              ( COMMAND-MODE, READ STRING )
  SHELL{               ( SET UP TO EVALUATE INTERPRET RESULT )
    INTERPRET          ( INTERPRET COMMAND )
  }SHELL
  IF ." OK"
  ELSE ." CAN'T INTERPRET"
  THEN
ELSE BUF 80 EXPECT      ( READ STRING INTO BUF )
  DECODE-STRING         ( GO DECODE IT )
THEN
```

Note: BUF is a buffer which has been previously allocated. DECODE-STRING is a word which has been previously compiled into the dictionary.

9.3. DISPLAY ASPECTS

Varying levels of screen support are provided on the P.T. The simplest form is encountered when interpretive (dumb) FORTH is executed. In this mode, the FORTH machine buys a 22 line scene from the terminal/console function (TCF) and manages it. The top line of the screen is reserved as a status line, and FORTH handles the remaining 22 lines on a scrolling basis exactly as on a terminal using traditional FORTH.

Many applications impose requirements which are not satisfied using this form of screen management. These requirements include:

- multi window support eg. function key display window
- simultaneous operation eg. user commands are echoed on the command line, while live data is displayed in it's window
- wrap-around display as opposed to scrolling eg. live data display is very readable using a wrap-around display. (On the other hand, displaying data out of a capture buffer is more effective when employing scrolling).

All of these capabilities are provided for when the Display Manager is invoked. However, added capability brings with it some measure of additional complexity: the necessity for the user program to initialize and specify which window display it is to go to.

Alternatively, but usually in addition to the mechanism just described, is the utilization of a set of commands generally referred to as the **M.** command group.

This means of screen output operates on display ram as if it were general purpose ram. The entire command group works with two variables: the display pointer **POINTER-M**, and the word containing the color/code set attribute: **ATTR-M**.

Consider the following example consisting of 4 windows:

```
( _____ )
( File      :DISPLAY.F)
( _____ )
(           X.25  Monitor  Test Manager Scenario
( _____ )
```

```
0  VARIABLE  DATA-WINDOW      ( Data display window identifier )
0  VARIABLE  HELP-WINDOW      ( Help display window identifier )
```

```
: INITIALIZE_SCREEN ( -- )
    ( Initialize display screen into 4 windows )
    ( Line 0      - 1 line status window )
    ( Line 1 to 16 - 16 line data window )
    ( Line 1 to 16 - 16 line help window )
    ( Line 17 to 19 - 3 line command window )
    ( Line 20 to 22 - 3 line function key window )
    ( The data window and help window )
    ( will be an alternate selection )
```

```
INIT_DISPLAY      ( Flush all former windows and
                   initialize new display )
```

```
1 16 BUY_WINDOW   ( Buy data window )
DUP DATA-WINDOW ! ( Store data window identifier )
POP_WINDOW        ( Open and display data window )
CLOSE_WINDOW      ( Close data window )
```

```
1 16 BUY_WINDOW   ( Buy help window )
DUP HELP-WINDOW ! ( Store help window identifier )
POP_WINDOW        ( Open and display help window )
GRN_FG PAINT      ( Set color of help window to green )
CLOSE_WINDOW      ( Close help window )
```

```
17 3 BUY_WINDOW   ( Buy command window )
DUP {FORTH-W} !   ( Store command window identifier )
POP_WINDOW        ( Open and display command window )
YEL_FG PAINT      ( Set color of command window to yellow )
CLOSE_WINDOW      ( Close command window )
```

```
3 INIT_FKEYS      ( Initialize 3 function key lines )
```

```
: POP_DATA      ( -- )      ( Open and display data window )
    DATA-WINDOW @ POP_WINDOW
;

: POP_HELP      ( -- )      ( Open and display help window )
    HELP-WINDOW @ POP_WINDOW
;

: POP_STATUS    ( -- )      ( Open and display status line window )
    STATUS-W POP_WINDOW
;

INITIALIZE_SCREEN

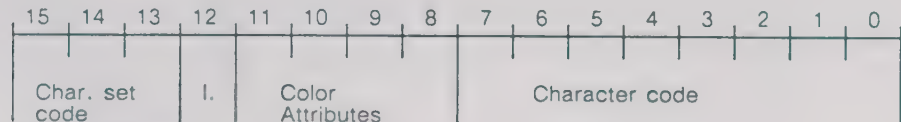
POP_STATUS
0 0 THERE
" This is the status line window " COUNT    BLU_BG W.TYPE_A

POP_DATA
3 20 THERE
W." This is the data window "

POP_HELP
2 10 THERE
W." This is the help screen "
4 10 THERE
W." Use POP_HELP to open and display this window "
6 10 THERE
W." Use POP_DATA to open and display data window "
```


DISPLAY RAM STRUCTURE.

The display ram consists of 32k bytes of RAM borrowed from the system ram and always resides at the top of the main CPU ram address space. Each screen location corresponds to a 16 bit location in ram. The low-order byte contains the character code, whereas the high-order byte contains the attributes.



Character Set:	Name	Code	Character Code
	ASCII	0	0x00 - 0xff
	EBCDIC	1	0x00 - 0xff
	HEX	2	0x00 - 0xff
	TTX	3	0x00 - 0xff
	JS8	4	0x00 - 0x35

I. = INVERSE VIDEO 0 NORMAL 1 inverse video

Color attributes: bits BACK FORE
 11 10 9 8

0 0	- -	black background
0 1	- -	blue background
1 0	- -	magenta background
1 1	- -	red background
- -	0 0	white foreground
- -	0 1	green foreground
- -	1 0	yellow foreground
- -	1 1	cyan foreground

9.4. DESIGNING A CONTROL LOOP

A control loop to manage a compiled program can be designed with relative ease. This control loop must appear at the top of the dictionary, i.e. all words called in the control loop must have been previously compiled into the dictionary. To illustrate the procedure, the following example will be used:

```

: RUN_APPLICATION      ( called manually or with " RUN_PROGRAM" )
  INITIALIZE           ( call INITIALIZATION ROUTINE-varies with application )
  EVENT_ON             ( enable event queue entry )
  BEGIN               ( start control loop )
    POP_EVENT          ( pop and event )
    DOCASE             ( start of case construct )
      CASE STR_IND     { KBD_DECODE } ( decode keyboard entry )
      CASE FK_IND      { FK_DECODE }  ( decode function key entry )
      CASE EOF_IND     { FRAME_DECODE } ( decode received frame )
      CASE 0           { WAIT_EVENT }  ( wait for another event )
    ENDCASE           ( end of case construct )
    DONE_FLAG @       ( fetch done flag )
  UNTIL              ( exit if set, otherwise return following begin )
;

```

The words INITIALIZE, KBD_DECODE, FK_DECODE, FRAME_DECODE and the variable DONE_FLAG are assumed to already exist in the dictionary.

9.5. PORT CONFIGURATION

Each port (test port direction) has its own configuration buffer. The default setting at power-up is BOP protocol with the interface mode set to monitor.

When the **PROTOCOL** command is executed, the specified configuration buffer(s) is (are) loaded with the default values. Following this step, the user may modify parameters which deviate from the default setting. The configuration buffers may be examined using the **.IBUF** command and then compared to the command buffer layout in the configuration section of the FORTH ENHANCEMENTS. The name of the configuration buffers are as follows:

PORT1_ACBUF, PORT1_BCBUF,
PORT2_ACBUF, PORT2_BCBUF,
PRINTER_CBUF, REMOTE_CBUF

EXAMPLE: **PORT1_ACBUF** **.IBUF**

Note: To obtain a readable configuration, the user should go to interpretive mode (enter **GOFORTH**)

A typical sequence may be as follows:

PORT1 BOP PROTOCOL	(protocol is BOP)
PORT1 9600 SPEED	(9600 bps)
PORT1 TODTE I/F_MODE	(interface mode is TO_DTE)
PORT1 CONFIG	(configure the hardware)
PORT1 RX_ON	(enable the receiver)

WARNING: TO CONFIGURE A PORT SPECIFY THE PROTOCOL mode **first**. This will assign all subsequent configuration commands to the dedicated PROTOCOL (e.g. BOP) configuration buffer.

9.6. TRANSMISSION ASPECTS

The command SEND_DATA has been provided to permit data buffers to be sent to the transmitters. (Refer to the SEND_DATA command in the FORTH ENHANCEMENTS section) The command requires the PORT ID, address of the data buffer and the attribute to be passed on the computational stack.

The data buffer structure is as follows:

```
WORDS 1-2    : Address where timestamp is to be written
WORD 3       : Byte count
BYTE7-N      : Data
```

The following examples illustrate the operations required:

Example 1: Transmitting canned messages

First, a data buffer must be allotted using the command 0 VARIABLE BUF1 80 ALLOT, where BUF1 is the variable name, and ALLOT allocates a function 80 bytes for a total of 84 bytes (The variable BUF1 occupies 4 bytes) NOTE: The allocation may not be done inside a colon definition.

A canned message will be defined in a literal string. (The string is stored in a literal buffer with the first character Position containing the character count)

```
" THE QUICK BROWN FOX JUMPED OVER THE LAZY DOG'S BACK 0123456789 CR LF CR "
```

During the initialization operation, the string must be transferred to BUF1 with a sequence like the one shown below to follow the literal string entry.

```
COUNT      ( FETCH THE COUNT )
DUP        ( DUPLICATE IT )
BUF1       ( GET ADDRESS OF BUFFER )
4+         ( STEP TO COUNT POSITION )
W!         ( STORE THE COUNT )
BUF1       ( GET ADDRESS AGAIN )
6+         ( STEP TO FIRST DATA POSITION )
SWAP       ( EXCHANGE WITH COUNT )
CMOVE      ( TRANSFER THE STRING )
```

The word which may be used to transmit the covered message is now defined. (PORT ID is TO_DTE_TX or TO_DCE_TX)

```
: TXCD      ( PORT ID -- )
  BUF1      NO_CRC SEND_DATA
;
```

Example 2: Transmitting user defined strings

We will define a data buffer BUF2 in the same way as Example 1.

```
0 VARIABLE BUF2 80 ALLOT
```

The user enters the string to be transmitted between quotes. Note that this operation results in the address of the string being placed on the computational stack. We will define a word TXUD which copies the string into BUF2 and initiates the transfer.

```
: TXUD          ( PORT ID/STRING ADDRESS -- )
  COUNT          ( FETCH THE COUNT )
  DUP            ( DUPLICATE IT )
  BUF2           ( GET ADDRESS OF BUF2 )
  4+             ( STEP TO COUNT POSITION )
  W!            ( STORE COUNT )
  BUF2           ( GET ADDRESS AGAIN )
  6+            ( STEP TO FIRST DATA POSITION )
  SWAP          ( EXCHANGE WITH COUNT )
  CMOVE         ( TRANSFER THE STRING )
  BUF2          ( GET ADDRESS )
  CRC           ( SPECIFY CRC 16 )
  SEND_DATA     ( SEND IT )
;
```

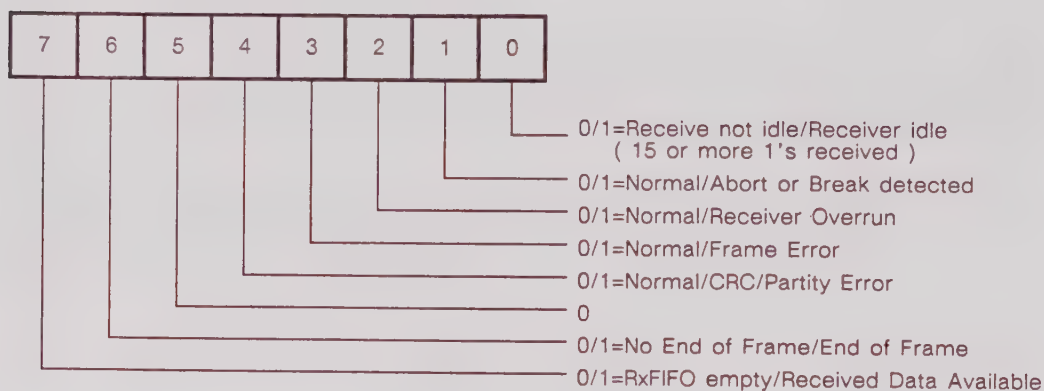
9.7. DATA RECEPTION ASPECTS

Received data and interface lead transitions are packaged into buffers. Indications containing pointers to these buffers are then sent to the application program.

Once the receivers have been enabled (using RX_ON or LEAD_REPORT_ON), the indications listed under RECEIVE DATA ENTRIES in the Forth Enhancements section could appear in the event queue. For received data, the following sequence would extract the relevant information:

```
: RX_FRAME      ( -- STATUS BYTE / ADDRESS / CHAR. COUNT / BLOCK COUNT /
                  END TS ADDR / START TS ADDR / PORT ID / 1 )
  EXTRACT_DATA  ( EXTRACT ADDRESS OF DATA BUFFER )
  UNPACK_DATA   ( EXTRACT REST OF INFORMATION )
;
```

The status byte is structured as follows:



The normal status is 0x40=End of Frame

- ADDRESS - is the address of the data buffer, whereas CHAR COUNT pertains to the number of characters.
- BLOCK COUNT - is the block sequence count of the received frame.
- END TS ADDR & - are the addresses which contain the timestamp for the beginning and end of frame conditions.
- START TS ADDR
- PORT ID - is one of the values from PORT DEFINITIONS.

Received interface lead transitions may be extracted using the following sequence:

```

: RX_LEAD          ( -- LEAD STATUS / LEAD CHANGE MASK / TX ADDR /
                   PORT ID / 0 )
  EXTRACT_LEADS    ( EXTRACT ADDRESS OF DATA BUFFER )
  UNPACK_DATA      ( EXTRACT REST OF INFORMATION )
;

```

The LEAD STATUS reflects the current status of all the interface leads (1=ON, 0=OFF). The bit positions correspond to the LIST OF INTERFACE LEADS in the FORTH ENHANCEMENTS section.

The LEAD CHANGE MASK indicates which lead underwent a transition. Once the lead has been identified through some mechanism like the CASE operation, the value of the lead can be quickly identified by performing an AND operation. If the result is 0, the lead is off, otherwise it is on.

TS ADDR is the address which contains the timestamp corresponding to the time at which the transition took place.

2. SIEMENS FORTH WORDLIST AND INDEX

LEGEND:

Types

W	Word
V	VARIABLE
C	CONSTANT
A	Address
(c)	like a constant

Source

fig	FORTH Interest Group
stand	standard, fig+forth79
siem	SIEMENS
siem-sys	SIEMENS system, recommended for general use

Format

Parameter stack description follows the format:
(value1/value2 -- value3)
where '/' should be read as 'under' and ' -- '
as 'leaves'. The stack can be visualized as
growing from bottom to top, with the last entry
being on top.

Symbols

addr	memory address
b	8 bit byte (i.e. high 24 bits zero)
c	7 bit ASCII character
f	boolean flag. 0 = false, non-zero = true
ff	boolean false flag = 0
n	16 bit signed integer number
u	32 bit unsigned integer
tf	boolean true flag = non-zero
d	32 bit signed integer number
/	and
	or
{ }	optional

WORD	Type	Source	Stack Description
.	W stand	(d --)	output number 3-5
."	W stand	(--)	outputs string 3-5
.4A	W siem	(value --)	output 4 ASCII bytes value 3-5
.H	W siem	(value --)	3-5
.HB	W siem	(value --)	output byte in HEX 3-5
.HH	W siem	(value --)	output halfbyte in HEX 3-5
.IBUF	W siem	(conf buf --)	displays 32 word buffer 4-54
.R	W fig	(d1/d2 --)	print value 3-5
.RAM	W siem	(--)	print info about dictionary space 3-6
.S	W siem	(--)	print top of stack non destructive 3-6
!	W stand	(d/addr --)	stores 32 bits of d at address 3-3
!!	W siem	(--)	Enables the cursor to be moved inside the commanded-windows to repeat previously given commands 6-2
!CSP	W fig	(--)	saves stack position in CSP 3-3
?	W stand	(addr --)	output contents of preceding VARIABLE 3-8
?*SEARCH	W siem	(target_addr/target_length/key -- next_addr/flag)	3-9
?COMP	W fig	(--)	issues error message if not compiling 3-9
?CSP	W fig	(--)	issues error message if csp check not OK 3-9
?DUP	W siem	(d1 -- d1)	duplicates value if non-zero 3-9
?ERROR	W fig	(f/n --)	issue error message if true 3-9
?EXEC	W fig	(--)	issues error message if not executing 3-9
?MATCH	W siem	(target_address/target_length/key_string addr -- flag)	matches key string and returns flag 3-9
?NUMBER	W siem	(pointer_to_text -- tf)	returns true if text is no. 3-9
?PAIRS	W fig	(n1/n2 --)	error message if n1 does not equal n2 3-9
?RANGE	W siem-sys	(value/min/max/string_addr -- value)	does range check 3-9
?RETURN_SET	W siem-sys	(-- flag)	returns true if SET_RETURN is active 4-43
?STACK	W fig	(--)	issues error message if stack out of bounds 3-9
?VERSION	W siem	(-- Revnumber / Versionnumber / Versionnumber / Address)	! 4-7
,	W stand	(d --)	stores 32 bit value on dictionary 3-5
;	W stand	(--)	terminates a colon definition 3-8
;AND	W fig	(--)	allows continuation of "making" definition 4-11
;NEXT	W siem	(--)	delimits action boundaries in key sequence 4-41
:	W stand	(--)	start compilable definition 3-7
"	W siem	(-- addr)	character string input 3-3
""	W siem	(-- addr/fk#)	4-39
'	W stand	(-- addr)	leaves parameter field address of word 3-3
""	W siem	(-- addr/fk#)	4-39
(	W stand	(--)	begin comment 3-3
(.)"	W stand	(--)	transmits in-line text to output device 3-3
(;CODE)	W fig	(--)	runtime part for ;CODE 3-3

(+LOOP)	W fig	(n --) runtime part for +LOOP	3-3
(CONST)	W 68kfig	(-- value) runtime code for CONSTANT	3-4
(DO)	W fig	(limit/index --) runtime part for DO	3-4
(DSTATUS)	V siem	(-- addr) Returns address of variable containing Data Recording Status	4-29
(ECA)	W siem	(--) compile time part for ENDCASE	3-4
(FIND)	W fig	(addr1/addr2 -- pfa/b/ta) if successful (addr1/addr2 -- ff) if not successful	3-4
(LOOP)	W fig	(--) runtime part of LOOP	3-4
(NEST)	W 68kfig	(--) codepointer for high level def.	3-4
(NFECPCBD)	W siem	(-- mask) returns mask for # of front end comm. processor boards	4-57
(NFLOPPY)	W siem	(-- mask) returns mask for # of floppy drives	4-26
(NFORTH)	W siem	(-- mask) returns mask for # of FORTH machines	5-4
(NUMBER)	W fig	(d1/addr1 -- d2/addr2) converts ASCII->bin to BASE	3-4
(NWINC)	W siem	(-- mask) returns mask for # of logical Winchester drives	4-26
(SET_RETURN)	W siem-sys	(--) runtime part of SET_RETURN	4-43
(TAKE_ERROR_RETURN)	W siem-sys	(-- 0) failed interpretation	4-43
(TAKE_RETURN)	W siem-sys	(-- 1) successful interpretation	4-43
(VAR)	V 68kfig	(-- addr) runtime part for VARIABLE	3-4
(W.")		(--)	4-32
[	W stand	(--) suspends compilation	3-23
[COMPILE]	W stand	(--) forces compilation of an immediate def.	3-23
]	W stand	(--) resumes compilation	3-23
{	W siem	(--) starts execution sequence if TRUE condition	4-10
{FORTH-W}	V siem	(-- addr)	4-34
}	W siem	(--) terminates the case construct	4-10
}ACTION	W siem	(--)	8-2
}SHELL		(-- flag) terminates shell & returns flag indicating error	4-43
}STATE	W siem	(--)	8-2
#	W stand	(d1 -- d2) generates double number d1	3-3
#"	W siem	(-- value) four byte as a literal	3-3
#>	W stand	(d -- addr/d) terminates numeric output conv.	3-3
#_LOOPS	V siem	(-- addr) general purpose temporary variable	4-5
#ELSE	W siem	(--) false part of #IFDEF	4-11
#ENDIF	W siem	(--) marks end of #IFDEF conditional compile	4-11
#IF		(-- state)	4-12
#IFDEF	W siem	(--) checks if identifier defined in dictionary	4-11
#IFNOTDEF	W siem	(--) checks if identifier defined in dictionary	4-12
#IS_FALSE	W siem	(state/flag --)	4-12
#IS_TRUE	W siem	(state/flag --)	4-12
#S	W stand	(d1 -- d2) generates ASCII text in output buffer	3-3
@	W stand	(addr -- d) leaves 32 bit contents of address	3-9
+	W stand	(d1/d2 -- sum) addition	3-4
+	W stand	(d/addr --) add value to VARIABLE	3-4
+-	W fig	(d1/d2 -- d) applies value of d2 to d1	3-4
+LOOP	W stand	loop with inc_value	3-5
-	W stand	(d1/d2 -- diff) difference of d1 -d2	3-5

-FIND	W fig	(-- pfa/b/ta) or false if not found read word from input and search dict.	3-5
-TRAILING	W stand	(addr/d1 -- addr/d2) suppress trailing blanks	3-5
*	W stand	(d1/d2 -- prod) leaves signed product of 2 numbers	3-4
*/	W stand	(n1/n2/n3 -- n4) multiplication, division	3-4
*/MOD	W stand	(n1/n2/n3 -- n4/n5) like */ plus remainder	3-4
/	W stand	(n1/n2 -- quot) divide	3-6
/MOD	W fig	(n1/n2 -- rem/quot) divide plus remainder	3-6
=	W stand	(d1/d2 -- f) true if d1 = d2	3-8
<	W stand	(d1/d2 -- f) true if d1 < d2	3-8
<<	W siem	(d1 -- d2) shift top of stack 1 bit left = (*2)	3-8
<#	W stand	(--) start pictured numeric conversion	3-8
<BUILDS	W stand	(--)	3-8
<CMOVE	W stand	(addr1/addr2/count --)	3-8
<<#	W siem	(d1/# of bits -- d2) Logical shift of a 32 bit value to the left by the specific number of bits.	3-8
>	W stand	(val1/val2 -- flag) true if val1>val2	3-8
>>	W siem	(d1 -- d2) shift top of stack one bit right = (/2)	3-8
>>#	W siem	(d1/# of bits -- d2)	3-8
>R	W stand	(d --)	3-8
>S	W siem	(n --) moves item from parameter stack to shell stack	4-44
0	C fig	(-- d)	3-6
0<	W stand	(d -- f) true if value < 0	3-6
0=	W stand	(d -- f) true if value = 0 invert truth flag	3-6
0BRANCH	W fig	(f --) branch if zero	3-6
1	C fig	(-- d)	3-6
1+	W stand	(d1 -- d1 +1) adds 1 to value on stack	3-6
10+	W siem	(d1 -- d1 +10) adds 10 to value on stack	3-6
12+	W siem	(d1 -- d1 +12) adds 12 to value on stack	3-6
14+	W siem	(d1 -- d1 +14) adds 14 to value on stack	3-7
16+	W siem	(d1 -- d1 +16) adds 16 to value on stack	3-7
18+	W siem	(d1 -- d1 +18) adds 18 to value on stack	3-7
2	C fig	(-- d)	3-7
2+	W stand	(d1 -- d2 +2) adds 2 to value on stack	3-7
2*	W stand	(n -- n*2) multiplies number on stack by 2	3-7
2/	W stand	(n -- n/2) divides number on stack by 2	3-7
20+	W siem	(d1 -- d1 +20) adds 20 to value on stack	3-7
22+	W siem	(d1 -- d1 +22) adds 22 to value on stack	3-7
24+	W siem	(d1 -- d1 +24) adds 24 to value on stack	3-7
2DROP	W siem	(d1/d2 --) drop 2 longs	3-7
2DUP	W siem	(d1/d2 -- d1/d2/d1/d2) duplicate top 2	3-7
2OVER	W siem	(a / b / c / d -- a / b / c / d / a / b)	3-7
2SWAP	W siem	(a / b / c / d -- c / d / a / b)	3-7
3	C siem	(-- d)	3-7
3+	W stand	(d1 -- d1 +3) adds 3 to value on stack	3-6
3DROP	W siem	(d1/d2/d3 --) drop 3 longs	3-7
3DUP	W siem	(d1/d2/d3 -- d1/d2/d3/d1/d2/d3) duplicate to 3	3-7
4	C siem	(-- d)	3-7
4+	W siem	(d1 -- d1 +4) adds 4 to value on stack	3-6

4*	W stand	(n -- n*4) multiplies number on stack by 4	3-7
4/	W stand	(n -- n/4) divides number on stack by 4	3-7
6+	W stand	(d1 -- d1 +6) adds 6 to value on stack	3-6
7FFF	C stand	(-- 0X7FFF) value of HEX 7FFF	3-7
8+	W siem	(d1 -- d1 +8) adds 8 to value on stack	3-6
ABORT	W stand	(--) start routine	3-9
ABS	W stand	(d -- d) returns the absolute value of d	3-9
ABT	C siem	(-- value) abort argument for SEND	4-81
ACTION{	W siem	(Ja oder nein --) code to be executed	8-2
ACTIVATION_IND	W siem	(-- value) ACTIVATION INDICATION OF LAYER 1	4-64
AGAIN	W stand	(addr/d --) loopback for 'infinite' loop	3-10
ALITBUF_	V siem	(-- addr) buffer for interpreted "	3-10
ALLOT	W stand	(d --) reserves dictionary space	3-10
ALLCTL	W siem	(d --) reserves dict. space by 32 bit long words	3-10
ALLTOW	W siem	(d --) reserves dict. space by 16 bit words	3-10
ALTER	W siem	(--) Definition of the dictionary word nnnn.	3-10
AND	W stand	(d1/d2 -- d3) leaves bitwise logical	6-1
ASCII	C siem	(-- value) ASCII constant	3-10
ASCII_TO_EBCDIC	C siem	(-- value)	4-36
ASTART_	W siem-sys	(-- addr) when 1, system loads file #ALOAD	4-7
ASYN	C siem	(-- value) ASYN constant	4-48
ASYN_TIME	W siem	(port/time --) select timeout for ASYN framing	4-51
ATTR-M	V siem	(-- addr) attribute for screen M	4-37
AUTO_CLEAR_OFF	W siem	(--)	4-31
AUTO_CLEAR_ON	W siem	(--) clears following line after CR is printed	4-31
AUTO_DISK_RPT	W siem	(-- addr) reports status after any disk operation	4-7
AUTO_REBOOT	V siem	(-- addr)	4-7
AUTONUM	V siem	(-- addr) control of autonumber option for 0X, 0 and 0B number input	3-10
AUTOSTART	W siem	(--) loads file ALOAD from disk DR0	3-10
B_CHANNEL	W siem	(-- value)	4-76
B1_CHANNEL	W siem	(-- value) B1 CHANNEL ID	4-64
B2_CHANNEL	W siem	(-- value) B2 CHANNEL ID	4-64
BACK	W fig	(addr --) resolve backward branch offset	3-10
BASE	V stand	(-- addr) current number base	3-10
BASEYEAR	W siem	(baseyear --) sets the base year	4-19
BCC	W siem	(port/value --) specify block check code	4-51
BCH_SWITCH	W siem	(portid / mode --) assigns the B1 channel and B2 channel across	4-62
BCH_VALID	W siem	(portid / B-chid -- / -- state) returns the state of the selected B channel, i.e. idle or active	4-62
BCHEXT	W siem	(portid / mode --) routes B channels to the BNC jack on the connector module	4-62
BEEP	W siem	(--) produces one beep	4-18
BEEP_DUR	V siem	(-- addr) returns addr. of variable used to set beeper	4-18
BEEP_OFF	W siem	(--) turns beeper off	4-18
BEEP_ON	W siem	(--) turns beeper on	4-18
BEEP_TONE		(-- addr) returns address of variables used	4-18
BEGIN	W stand	(-- addr/d) marks start of sequence	3-10

BEGIN_KEY_SEQ	W siem	(--) starts definition of n-element function key sequence	4-40
BIN	W siem	(--) set BASE to binary	3-10
BL	C fig	(-- value) blank value	3-11
BLANKS	W stand	(addr/count --) fills area of memory with blanks	3-11
BLK	V stand	(-- addr) ->block# currently interpreted	3-11
BLK_BG	C siem	(-- value) black background	4-36
BLU_BG	C siem	(-- value) blue background	4-36
BLU_FG	C siem	(-- value) blue foreground	4-36
BOF_IND	C siem	(-- value) beginning of frame indication	4-83
BOP	C siem	(-- value) BOP protocol constant	4-48
BPCH	W siem	(port/value --) number of bits per character	4-50
BRANCH	W fig	(--) unconditional high level branch	3-11
BSC_A	C siem	(-- value) BISYNC ASCII protocol	4-48
BSC_E	C siem	(-- value) BISYNC EBCDIC protocol	4-48
BTSTK	C siem	(-- value) return end of stack address	3-11
BUILDS	W fig	(--) Defined Word,	3-8
BUS_MODE	W siem	(portid / mode --) specifies the access mode for Layer1 device	4-58
BUY_KEYS	W siem	(port/key_group --) claim ownership of keys	4-42
BUY_WINDOW	W siem	(start_row/no_of_rows -- wd) creates new window	4-30
C!	W stand	(b/addr --) store byte	3-11
C,	W stand	(b --) , for bytes	3-11
C@	W stand	(Addr -- b)	3-11
C-B	V siem	(-- addr) address of capture buffer	4-45
C-B-SIZE	V siem-sys	(-- addr) size in bytes of capture buffer	4-45
C/I_MON	W siem	(portid / mode --) turns the Layer1 monitor on or off	4-61
C/L	C fig	(-- value) characters per line	3-11
C=	W stand	(b1/b2 -- f) like=, but for byte comparison	3-11
CAPTURE	W siem	(addr/count -- captured?) capture data to RAM	4-45
CASE	W siem	(--) for case construct	4-9
CB_VAR_INIT	W siem-sys	(id --)	4-44
CDOWN	C siem	(-- value) down cursor key	4-91
CF	C siem	(-- value) control function key group	4-91
CF1	C siem	(-- value) control function key group 1	4-91
CF2	C siem	(-- value) control function key group 2	4-91
CF3	C siem	(-- value) control function key group 3	4-91
CF4	C siem	(-- value) control function key group 4	4-91
CF5	C siem	(-- value) control function key group 5	4-91
CF6	C siem	(-- value) control function key group 6	4-91
CF7	C siem	(-- value) control function key group 7	4-91
CF8	C siem	(-- value) control function key group 8	4-91
CFA	W fig	(pfa -- cfa) converts parameter field address	3-11
CHANGE	W siem	(-- PFA) Changes the meaning of a word into another word.	6-2
CHARS/LINE	W siem	(port/value --)	4-52
CHECK	W siem	(port/lead --) check interface lead	4-13
CHOME	C siem	(-- value) home cursor key	4-91
CLEAN_WINDOW		(--) erases text and paint of current window	4-33
CLEAR_ROW	W siem	(row --) clears text from one row of window	4-33

CLEAR_TEXT	W siem	(--) clears all text from window	4-33
CLEFT	C siem	(-- value) left cursor key	4-91
CLOCK_DSP	W siem	({YES NO} --) turns on/off clock display on status line	4-19
CLOCK_SOURCE	W siem	(portid / mode --) specifies the clock source for interface and attached periphérie	4-59
CLOSE	W siem	(--) close file which was last opened	4-23
CLOSE_ALL	W siem	(--) close all files	4-23
CLOSE_FILE	W siem	(truncate_flag/file_desc --) close file	4-23
CLOSE_WINDOW	W siem	(--) cancel an OPEN_WINDOW	4-32
CMOVE	W stand	(addr1/addr2/count --) block move	3-11
COLD	W fig	(--) start routine	3-11
Cols	W siem	(-- n) returns number of columns in a window	4-35
COME_BACK	W siem	(--) Bad trace of last Interrupt is displayed.	6-3
COMMAND_IND	C siem	(-- value) multi FORTH command indication	4-83
COMPARE	W siem	(source/dest --) compares files from source	4-25
COMPILE	W stand	(--) word following COMPILE is compiled	3-12
CONF_BUF_ASYNC	V siem	(-- addr) default configuration buffer ASYNC	4-57
CONF_BUF_BOP	V siem	(-- addr) default configuration buffer BOP	4-57
CONF_BUF_BSCA	V siem	(-- addr) default configuration buffer BSCA	4-57
CONF_BUF_BSCE	V siem	(-- addr) default configuration buffer BSCE	4-57
CONF_BUF_CCS#7	V siem	(-- addr) Default configuration buffer for CCS#7 monitor and simulator.	4-57
CONF_BUF_COP	V siem	(-- addr) default configuration buffer COP	4-57
CONF_BUF_ISOC	V siem	(-- addr) default configuration buffer ISOC	4-57
CONFIG	W siem	(port --) configure interface port	4-57
CONNECTOR	W siem	(-- connector type) returns the local connector type	4-8
CONNECTOR1	W siem	(-- connector-type) returns local connector type of the interface	4-8
CONNECTOR2	W siem	(-- connector-type) returns local connector type of the interface	4-8
CONSTANT	W stand	(d --) generate a constant:val CONSTANT name	3-12
CONTEXT	V stand	(-- addr) pointer to voc. to be searched first	3-12
CONV	V siem	(source/count/dest/code -- 1 count/0)	4-92
COP	C siem	(-- value) COP protocol	4-48
COPY	W siem	(source/dest --) copies file from source diskette	4-24
COUNT	W stand	(addr1 -- addr2/n) prepare for TYPE	3-12
CR	W stand	(--) generates a carriage return & line feed	3-12
CRC	C siem	(-- value) parameter for crc calculation	4-81
CRC_16	C siem	(-- value) CRC_16 BCC type	4-51
CRC_CALC	W siem	(addr/count/crc_type -- MSB/LSB) calculates CRC for CRC_16 and CRC_CCITT	4-92
CRC_CCITT	C siem	(-- value) CRC_CCITT BCC type	4-51
CRC_ERR	C siem	(-- value) parameter for crc error generation	4-81
CRC4_FRAMING	C siem	(-- value) Parameter for PRI_Framing	4-75
CREATE	W fig	(--) create dictionary header	3-12
CREATE_FILE	W siem	(filename/access/size/type -- file_desc)	4-22
CRIGHT	C siem	(-- value) right cursor key	4-91
CSP	V fig	(-- addr) storage for CSP for compiler security	3-12
CUP	C siem	(-- value) up cursor key	4-91

CUR_A	W siem	(row/col --) position cursor	4-37
CUR_DEV	W siem	(-- current_disk_device)	4-21
CUR_P	W siem	(-- row/col) report cursor position	4-38
CURRENT	V stand	(-- data_addr) pointer to current work vocabulary	3-12
CURSOR	C siem	(-- value) cursor key group	4-42
CURSOR_IND	C siem	(-- value) event indication for cursor key entry	4-83
CURSOR_OFF	W siem	(--) disables display of block cursor	4-31
CURSOR_ON	W siem	(--) enables display of block cursor	4-31
CVT_H_STR	W siem-sys	(pointer_to_text -- flag) convert HEX literal string	3-12
CYA_BG	C siem	(-- value) cyan background	4-36
CYA_FG	C siem	(-- value) cyan foreground	4-36
D.	W stand	(d --) same as .	3-13
D.R	W stand	(d/n --) output value with field width	3-13
D+-	W fig	(d1/n -- d2) same as +-	3-13
D_CHANNEL	C siem	(-- value) parameter for PRI_ROUTE. Kanatype.	4-76
D_INIT	W siem	(" filename/addr --) opens designated file	4-29
D_NEWEST	W siem	(-- {length/addr null}) returns length & address of newest block of data	4-29
D_NEXT	W siem	(-- {length/addr null}) returns length & address of next block of data	4-30
D_OLDEST	W siem	(-- {length/addr null}) returns length & address of oldest block of data	4-30
D_PREVIOUS	W siem	(-- {length/addr null}) returns length & address of previous block of data	4-30
D_STOP	W siem	(--) closes data file & terminates data playback	4-30
D4_FRAMING	C siem	(-- value) parameter for PRI_FRAMING	4-24
DABS	W stand	(d -- ud) leaves absolute value of double number	3-13
DCD_CONTROL	W siem	(port/id --) Selects carrier detect control mode	4-50
DEACTIVATION_IND	W siem	(-- value) deactivation indication of layer 1	4-64
DECIMAL	W stand	(--) set BASE to decimal	3-13
DECODE	W siem	(PFA/d --)	6-1
DECOMP	W siem	(--) Decomposes and prints the dictionary word	6-1
DECOMP.H	W siem	(--) Decomposes and prints the dictionary word nnnn with memory address in HEX.	6-1
DECOMP_H	W siem	(--) Decomposes and prints the dictionary word nnnn horizontally.	6-1
DEFINITIONS	W stand	(--) specify work vocabulary	3-13
DICTLIMIT	V fig68k	(-- addr) get address of dictionary limit	3-13
DIGIT	W fig	(c/n1 -- n2/ff) converts ASCII char. to binary (c/n1 -- ff) else	3-13
DIO	C siem	(-- value) disk I/O function number	4-89
DIR	W siem	(filename --) gets short directory listing	4-24
DIRL	W siem	(filename --) gets long directory listing	4-24
DISABLE_BCH	W siem	(portid / B-chid --) turns off the selected B channel	4-61
DISK_BUF	V siem	(-- addr) 1024 byte array	4-26
DISPLAY_KEY	W siem	(string addr/fk# --)	4-39
DISPLAY_KEY_A	W siem	(string addr/fk#/color --)	4-39
DISTABLE	W siem	(-- addr) returns address of display table	4-35
DLITERAL	W fig	same as LITERAL	3-13

DO	W stand	(d1/d2 --) begins sequence with repetitive exec. (addr/d --) compile	3-13
DOCASE	W siem	(--) starts the case construct	4-9
DOER	W fig	(--) defines a word vectorable by using MAKE DOES>	4-11
DOES>	W stand	(--)	3-8
DP	V fig	(-- addr) pointer to next free dict. position	3-14
DPL	V stand	(-- addr)	3-14
DR0	W fig	(--) selects disk drive 0	3-14
DR1	W fig	(--) selects disk drive 1	3-14
DREC_OFF	W siem	(--)	4-29
DREC_ON	W siem	(filename/size/access --)	4-29
DROP	W stand	(d --) drop number from stack	3-14
DROP_AND_INSERT	C siem	(-- value)4-68	4-75
DROP_WINDOW	W siem	(window_descriptor --)removesdesignate window	4-31
DSM	C siem	(-- value) data stream merger function number	4-89
DSM_ERR_IND	W siem	(-- value) disk error indication	4-83
DSM_RECORD?	W siem	(--) Returns the recording status of the DSM.	4-29
DSMBUF	W siem	(-- addr)	4-7
DSMOVRN	C siem	(-- value) data recording overrun error indication	4-83
DUMP	W stand	(addr/n --) dumps memory to screen	3-14
DUP	W stand	(d -- d/d) duplicates value on stack	3-14
EBCDIC	C siem	(-- value) EBCDIC constant	4-36
EBCDIC_TO_ASCII	C siem	(-- value)	4-92
EDF	C siem	(-- value) editor function number	4-89
EDIT	W siem	(--) calls up EDITOR	4-6
EDITBUF	W siem	(-- addr) returns starting address of Editor buffer	4-6
EDITF	W siem	(filename --) call EDITOR with FILENAME	4-6
EDITM	W siem	(filename --) call EDITOR	4-6
ELSE	W stand	(addr1/d1 -- addr2/d2)	3-14
EMIT	W stand	(c --) output top of stack, ASCII	3-14
ENABLE_BCH	W siem	(portid / B-chid --) turns on the selected B channel	4-61
ENCLOSE	W fig	(addr1/c --) parse string	3-14
ENDCASE	W siem	(--) terminates the case construct	4-10
ENDIF	W fig	(addr/d --) same as THEN to end IF	3-14
ENDRAM	W siem	(-- addr) returns address of first byte past end of local RAM	4-9
ENDRAM_M	W siem	(-- addr) returns address of first byte past end of main CPU RAM	4-9
ENDRAM_1	W siem	(-- addr) returns address of first byte past end of CPU1 RAM	4-9
ENDRAM_2	W siem	(-- addr) returns address of first byte past end of CPU2 RAM	4-9
ENDRAM_3	W siem	(-- addr) returns address of first byte past end of CPU3 RAM	4-9
ENDRAM_4	W siem	(-- addr) returns address of first byte past end of CPU4 RAM	4-9
ENDRAM_5	W siem	(-- addr) returns address of first byte past end of CPU5 RAM	4-9

ENDRAM_6	W siem	(-- addr) returns address of first byte past end of CPU6 RAM	4-9
ENDRAM_7	W siem	(-- addr) returns address of first byte past end of CPU7 RAM	4-9
ENDRAM_8	W siem	(-- addr) returns address of first byte past end of CPU8 RAM	4-9
EOF_COUNT	W siem	(port/value --) configures receiver to force end of frame condition	4-51
EOF_IND	C siem	(-- value) end of frame indication	4-83
EOL	W siem	(-- value) returns value for END OF LINE character	4-53
ERASE	W stand	(addr/d --) fill with 0 x 0	3-14
ERROR	W fig	(n --) executes error and restart of system	3-14
ESF_FRAMING	C siem	(-- value) USA-Superframe	4-75
EVEN	W siem	(port --) even Parity	4-76
EVEN_PAR	W siem	(port --) select even parity	4-50
EVEN_PAR_BIT	W siem	(char -- even_parity_bit_of_char)	4-92
EVENT_OFF	W siem	(--) disable event queue entry	4-83
EVENT_ON	W siem	(--) enable event queue entry	4-83
EXEC	W siem	(filename --) read and compile file	4-5
EXEC_ERR	V siem-sys	(-- tf) returns error flag	4-5
EXECB	W siem	(filename --) load binary file	4-5
EXECUTE	W fig	(addr --) executes definition of addr. on stack	3-15
EXECUTE_KEY	W siem	(fk# --)	4-40
EXECUTE_WORD	W siem	(string addr --) executes word if in dictionary	4-7
EXIT	W siem	(--) un-nest, exit inner interpreter	3-15
EXIT_SHELL	W siem	(--) exits from the shell	4-44
EXPECT	W stand	(addr/count --) text input from terminal	3-15
EXT_CLK	W siem	(port --) external clocking	4-49
EXTENDED_BUS	C siem	(-- value) extended bus operation	4-64
EXTRACT_BOF_TS	W siem	(-- beg_of_frame_time_stamp)	4-85
EXTRACT_DATA	W siem	(-- data_buffer_addr)	4-85
EXTRACT_DSM_ERR	W siem	(-- error_value)	4-87
EXTRACT_DSMMOVRN	W siem	(-- seq_#/R_x_ID) Extracts receiver stream id and sequence number of received block when DSM overrun.	4-87
EXTRACT_ERROR	W siem	(-- function_no/error)	4-87
EXTRACT_FTH_DATA	W siem	(-- d1/d2/.../#_of_d's)	5-3
EXTRACT_KEY	W siem	(-- functions_or_cursor_key_value)	4-85
EXTRACT_LEADS	W siem	(-- addr)	4-85
EXTRACT_PRI	W siem	(-- Adresse) extracts the address of the PRI indication buffer	4-75
EXTRACT_TIMER	W siem	(-- timer_no)	4-85
F_REMOVE	W siem	(filename --) removes file with force	4-25
FAST_M	W siem	(port --)	4-48
Fcol	W siem	(-- column) returns first column of window	4-35
FCOLORS	W siem	(-- addr)	4-35
FDBASEYR	W siem	(-- baseyear) returns base year associated with most recently opened, closed, or located file.	4-24
FDES	V siem-sys	(-- addr) contains file descriptor of file last opened (EXEC,EXECB,SAVEB)	4-22

FDSIZE	W siem	(-- file_size) returns size of last file	4-25
FDSTATUS	W siem	(-- status) returns status of last disk operation	4-25
FDTYPE	W siem	(-- file_type) returns file type	4-25
FEF_1	C siem	(-- FORTH_#1's_id Current_FORTH's_id)	5-3
FEF_2	C siem	(-- FORTH_#2's_id Current_FORTH's_id)	5-3
FEF_3	C siem	(-- FORTH_#3's_id Current_FORTH's_id)	5-3
FEF_4	C siem	(-- FORTH_#4's_id Current_FORTH's_id)	5-3
FEF_5	C siem	(-- FORTH_#5's_id Current_FORTH's_id)	5-3
FEF_6	C siem	(-- FORTH_#6's_id Current_FORTH's_id)	5-3
FEF_7	C siem	(-- FORTH_#7's_id Current_FORTH's_id)	5-3
FEF_8	C siem	(-- FORTH_#8's_id Current_FORTH's_id)	5-3
FEF_COLD	W siem	(FORTH# --) causes FORTH machine to perform COLD	5-2
FEF_DO_WORD	W siem	(" cmds"/FORTH# --) executes as if typed on keyboard	5-2
FEF_GOFORTH	W siem	(FORTH# --) causes FORTH machine to perform GOFORTH	5-2
FENCE	V fig	(-- addr) points to protected dict. location, all lower entries are protected against FORGET	3-15
FFFFFFFFE	C siem	(-- OXFFFFFFFFE) mask for even number	3-15
FILE_NAME	V siem	(-- addr) general purpose 16 byte array	4-16
FILES	W siem	(-- filename_specifiers) specifies the files names to be compiled	6-2
FILL	W stand	(addr/quant/b --)	3-15
FILLW	W siem	(addr/quant/w --) fill for 16 bit units	3-15
FIND	W siem	(--) Finds and prints all definitions that contain the dictionary word nnnn.	6-1
FIND_WORD	W siem-sys	(test_addr -- pfa/b/ff) OK (test_addr -- ff) BAD	4-7
FirstRow	W siem	(-- row)	4-34
FK_IND	C siem	(-- value) function key indication	4-83
FKM-LINES	V siem-sys	(-- addr) returns address of variable	4-29
FKM-SCENE	V siem-sys	(-- addr) contains RAM addr. of beginning display	4-39
FKM-WINDOW	W siem	(-- addr_of_the_window_descriptor)	4-40
FLD	V fig	(-- addr) presently unused	3-15
FORGET	W stand	(--) deletes definition and subsequent entries	3-15
FORMAT	W siem	(--) format currently selected disk	4-24
FORMATWD	W siem	(LS0/LS1/ -- -/LSn/#LS --) formats and logically partitions Winchester Disk	4-28
FORTH	W stand	(--) name of primary vocabulary	3-15
FROM	W siem	(--) Defines the cell aaaa in the dictionary word nnnn to be altered.	6-1
FTH	C siem	(-- value) function number for this FORTH func.	4-89
GEN_BPVL	C siem	(-- value) pri bipolar violations generated	4-76
GEN_CRCL	C siem	(-- value) pri crc errors generated	4-76
GEN_FERR	C siem	(-- value) pri frame errors generated	4-76
GEN_MERR	C siem	(-- value) pri multiframe errors generated	4-76
GET_ATTR	W siem	(-- value) get present display attributes	4-36
GET_I/F	W siem	(port -- physical_interface_type)	4-47
GET_PRI_I/F_MODE	W siem	(pri_pid -- mode) returns the PRI interface mode.	4-66
GET_TIME	W siem	(-- ss/mi/hh/dd/mo/yy) reads real-time clock	4-19

GET_TS	W siem	(timestamp_addr --)	4-19
GET_TSTAMP	W siem	(timestamp_addr -- μ ss/mss/ss/mi/hh/dd/mo/yy) converts timestamp to output format	4-20
GET_TSTAMP_MICROW	siem	(timestamp_addr -- μ ss/ss/mi/hh/dd/mo/yy) unpacks timestamp to microsec	4-20
GET_TSTAMP_MILLI	W siem	(timestamp_addr -- mss/ss/mi/hh/dd/mo/yy) unpacks timestamp to milliseconds	4-20
GOFORTH	W siem	(--) do warm start	3-15
GRN_BG	C siem	(-- value) green background	4-36
GRN_FG	C siem	(-- value) green foreground	4-36
H0_CHANNEL	C siem	(-- value) H0-Channel	4-76
H11_CHANNEL	C siem	(-- value) H11-Channel	4-76
H12_CHANNEL	C siem	(-- value) H12-Channel	4-76
HERE	V stand	(-- addr) leaves addr. of next available location	3-15
HEX	W stand	(--) sets numeric conversion base to HEX	3-15
HEX_LINE	W siem	(source_addr/count/offset--updated_source_addr) writes HEX output to POINTER-M	4-37
HEXSET	W siem	(-- value)	4-36
HIGH_IMP	C siem	(-- value) pri high impedance	4-76
HILITE_FIELD	W siem	(fk#/begin/end --)	4-39
HILITE_KEY	W siem	(fk# --)	4-39
HLD	V fig	(-- addr) points to last character of text placed into PAD by pictured output conversion	3-15
HLITBUF	V siem-sys	(-- addr) get address of the HEX literal storage	3-16
HOLD	W stand	(c --) stores an ASCII character to HLD + 1	3-16
HPRIO	C siem	(-- value) high priority access	4-64
I	W stand	(-- d) gets index of current DO LOOP, +LOOP	3-16
I/F_ENABLE	W siem	(port/Lead_id --) enables control lead messages	4-82
I/F_IND	C siem	(-- value) control lead transition	4-83
I/F_MODE	W siem	(port/mode --) specify interface mode	4-47
I/F_TYPE	W siem	(port/physical_interface_type --)	4-48
I_MODE	V siem	(-- addr) current interface mode	4-47
IC	C siem	(-- d) control V.11	4-13
ICS	C siem	(-- d) clear to send	4-13
ID.	W fig	(addr --) prints definition's name field address	3-16
IDLE	C siem	(-- value) pause word	4-76
IDM	C siem	(-- d) data set ready	4-13
IF	W stand	(f --) for IF ENDIF, IF ELSE THEN, etc.	3-16
IF_FILL	W siem	(port/value --) specify interframe fill condition	4-50
II	C siem	(-- d) indication V.11	4-13
IIC	C siem	(-- d) ring indicate	4-13
IIS	C siem	(-- d) terminal in service	4-13
ILL	C siem	(-- d) local loopback	3-16
IMMEDIATE	W stand	(--) marks most recent definition; at compile time will be executed rather than compiled	3-16
IN	V fig	(-- addr) byte offset for TIB	3-16
INIT_CAPTURE	W siem	(wrap_flag --) initialize capture buffer	4-44
INIT_DISPLAY	W siem	(--) initializes display manager to startup	4-31
INIT_FKEYS	W siem	(n --) initializes function key manager	4-39
INP_TERM	W siem	(char --) terminal entry termination character	4-42
INTERFACE	W siem	(-- interface-type) returns local Interface type	4-8

INTERFACE1	W siem	(-- interface-type) returns local Interface type of the interface	4-8
INTERFACE2	W siem	(-- interface-type) returns local Interface type of the interface	4-8
INTERPRET	W stand	(--) outer text interpreter	3-16
INTO	W siem	(PFA --) Replaces a word nnnn with the word ssss.	6-2
IP0 .. IP8	W siem	(-- addr)	4-8
IR	C siem	(-- d) receive V.11	4-13
IRR	C siem	(-- d) carrier detect	4-13
IRS	C siem	(-- d) request to send	4-13
ISOC	C siem	(-- value) ISOC protocol	4-48
ISQ	C siem	(-- d) signal quality	4-13
ISR	C siem	(-- d) data signal rate select to DCE	4-13
ISS	C siem	(-- d) select DCE standby facility	4-13
IT	C siem	(-- d) transmit V.11	4-13
ITR	C siem	(-- d) data terminal ready	4-13
J	W fig	(-- d) leaves outer loop index in double-nested DO LOOP	3-16
JS8	C siem	(-- value) JIS8 code	4-36
KBD	C siem	(-- value) entire keyboard group	4-42
KBYTE	W siem	(n -- nkbyte) leaves n *1024 on the stack.	3-16
KEY	W stand	(-- c) ASCII value of next key struck on keyboard	3-16
KILL_KEY	W siem	(fk# --)	4-40
L!	W siem	(value/addr --) same as !, store 'long'	3-17
L@	W siem	(addr -- value) same as @, fetch long	3-17
LAST_IPC	V siem	(-- addr) address of last popped IPC message	4-84
LATEST	V fig	(-- addr) leaves name field addr. of latest word	3-17
LB_ALTER	C siem	(-- value) loopback alternative	4-64
LB_BOTH	C siem	(-- value) loopback on both channels	4-64
LB_DISABLED	C siem	(-- value) loopback disabled	4-64
LEAD_DISPLAY_OFF	W siem	(port --) disable display of lead status	4-13
LEAD_DISPLAY_ON	W siem	(port/addr --) enables display of lead status	4-13
LEADING_PAD	W siem	(port/value --) specify leading pad character	4-51
LEAVE	W fig	(--) forces termination of DO-LOOP	3-17
LENGTH0	C siem	(-- value) pri line length 0-220 feet	4-76
LENGTH2	C siem	(-- value) pri line length 440-655 feet	4-76
LFA	W fig	(pfa -- lfa) converts a parameter field address into a link field address	3-17
LINES/PAGE	W siem	(port/value --)	4-52
LIT	W fig	(-- d) runtime part of LITERAL	3-17
LITERAL	W stand	compiler word	3-17
LOAD_FILES		(filename_specifiers --)	6-2
LOCAL	C siem	(-- value) local port id	4-3
LOCATE	W siem	(--) Locates the word nnnn and displays the source filename in where the word is located.	6-1
LOCATE_FILE	W siem	(" filename" -- 0 1) Locates the specified file from the filesystem	4-24
LOCK_WINDOW	W siem	(window --) locks specified window	4-34
LOOP	W stand	(addr/d --) for DO LOOP, etc. loops	3-17
LOOPBACK	W siem	(portid / mode --) mirrors either B1 channel, B2 channel or both	4-62

LPRI0	C siem	(-- value) LOW PRIORITY ACCESS	4-64
LRC	C siem	(-- value) LRC BCC type	4-51
LRC_EVEN	W siem	(addr/count -- even_LRC)	4-92
LRC_ODD	W siem	(addr/count -- odd_LRC)	4-92
LRC8	W siem	(addr/count -- LRC8) calculates & returns the LRC8	4-92
M.	W siem	(value/offset --) outputs a number at POINTER-M	4-37
M.HB	W siem	(value/offset --) outputs ASCII value at POINTER-M	4-37
M.HH	W siem	(value/offset --) outputs HEX value at POINTER-M	4-37
M.MOVE	W siem	(string addr/offset --) output string at POINTER-M	4-37
M.TYPE	W siem	(offset/string addr/count --) outputs string at POINTER-M	4-37
M*	W siem	(d1/d2 -- d) leaves signed product of 2 numbers	3-17
M/	W fig	(d/n1 -- n2/n3)	3-17
M/MOD	W fig	(ud1/u2 -- u3/ud 4)	3-17
MAG_BG	C siem	(-- value) magenta background	4-36
MAG_FG	C siem	(-- value) magenta foreground	4-36
MAIL	W siem	(function_no/buffer)	4-88
MAIL_WAIT	W siem	(function_no/buffer) mails buffer, waits for response	4-88
MAKE	W fig	(--) vectors doer-name word to FORTH code	4-24
MAKE_FS	W siem	(volume --) make file system	4-24
MAKE_KEY	W siem-sys	(--) assigns multiple actions to a key	4-40
MAX	W stand	(d1/d2 -- max) leaves greater of 2 numbers	3-17
MBOX-0	W siem	(-- addr) variable use by application software	4-7
MBOX-1	W siem	(-- addr) variable use by application software	4-7
MBOX-2	W siem	(-- addr) variable use by application software	4-7
MBOX-3	W siem	(-- addr) variable use by application software	4-7
MBOX-4	W siem	(-- addr) variable use by application software	4-7
MBOX-5	W siem	(-- addr) variable use by application software	4-7
MBOX-6	W siem	(-- addr) variable use by application software	4-7
MBOX-7	W siem	(-- addr) variable use by application software	4-7
MBOX-8	W siem	(-- addr) variable use by application software	4-7
MBOX-9	W siem	(-- addr) variable use by application software	4-7
MBSTATUS	W siem	(-- o buffer_addr)	4-89
MENU	W siem	(--) returns system to main menu page	4-5
MES_BUF	W siem	(-- 0 buffer_addr) 32_word buffer address returned or 0 if none available	4-89
MF	W siem	(--) identifies the version of operating system.	5-3
MFTH	C siem	(-- Main FORTH's_id)	5-3
MIN	W stand	(d1/d2 -- min) leaves smaller of 2 numbers	3-18
MOD	W stand	(n1/n2 -- mod) leaves remainder of n1 /n2	3-18
MON	C siem	(-- value) monitor port id	4-47
MOVE	W stand	(addr1/addr2/n --) block word move	3-18
MPH_ERROR_RESPONSE	W siem	(portid --) sets the Layer1 device into the deactivated state	4-60
MRK	C siem	(-- value) MARK parameter for SEND_DATA	4-81
MYPORT	W siem	(-- id of port a and port b for FORTH machine)	4-3
MYPORTA	W siem	(-- id of port a for that FORTH machine)	4-3
MYPORTB	W siem	(-- id of port b for that FORTH machine)	4-3
MYRX_CH		(-- TO_DCE rx stream id/TO_DTE rx stream id)	4-86

N->S	W siem	(addr_of_last_char/number -- addr_of_first_char)	3-18
NEGATE	W fig	(d1 -- d2) leaves the two's complement of number	3-18
NEW_STATE	W siem	(state_number --) Another way to change the state	8-2
NEWEST	W siem	(-- count/addr) newest captured frame	4-45
NEXT	W siem	(-- count/addr) next captured frame	4-45
NFA	W fig	(pfa -- nfa) convert parameter field address to its name field	3-18
NFECPCBD	W siem	(-- number_of_FECPC_boards) returns # of FECPC boards	4-57
NFLOPPY	W siem	(-- number_of_floppy_drives)	4-26
NFORTH	W siem	(-- number_of_FORTH_machines) returns # of FORTH machines	5-4
NN_EMULATION	C siem	(-- value) Parameter für PRI_I/F_Mode: Network/Network	4-75
NO	C siem	(-- 0) constant with a value of 0	3-18
NO_BCC	W siem	(-- n) returns value used by BCC	4-51
NO_PAR	W siem	(port --) select no parity	4-50
NO_SCROLL_DISABLE	W siem	(--) disables NOScroll key on keyboard	4-33
NO_SCROLL_ENABLE	W siem	(--) enables NOScroll key on keyboard	4-33
NO_TRUNCATE	C siem	(-- d) file to be truncated	4-22
NORM_M	W siem	(port --)	4-48
NOTHING	W siem	(--) Do nothing word used by DOER	4-11
NRZ	W siem	(port --) non return to zero	4-49
NRZI	W siem	(port --) non return to zero inverted	4-49
NRZIC	W siem	(port --) non return to zero inverted with clock	4-49
NT		(-- value) NETWORK	4-64
NU_EMULATION	C siem	(-- value) Parameter for PRI_I/F_Mode: Network/User	4-75
NULLIFY	W siem	(--) Changes the definition of the dictionary word nnnn to do nothing.	6-2
NULL_TERM	W siem	(addr/"string" --) terminates string and stores at address	3-18
NUMBER	W stand	(addr -- d) convert string to value by BASE	3-18
NWINC	W siem	(-- number_of_logical_Winchester_drives)	4-26
OCTAL	W siem	(--) sets number BASE to octal	3-18
ODD		(-- value) Ungerade	4-76
ODD_PAR	W siem	(port --) select odd parity	4-50
OFF	C siem	(-- 0) constant with a value of 0	3-19
OHM_120	C siem	(-- value) Parameter for PRI_IMPEDANCE	4-76
OHM_75	C siem	(-- value) Parameter for PRI_IMPEDANCE	4-76
OLDEST	W siem	(-- count/addr) oldest frame in capture buffer	4-45
ON	C siem	(-- 1) constant with a value of 1	3-19
OPEN_FILE	W siem	(filename/access -- file_desc.) open file	4-23
OPEN_RX	W siem	(port/beg_of_file_indications? --)	4-77
OPEN_WINDOW	W siem	(wd --) enables further output to be placed in this window	4-32
OR	W stand	(d1/d2 -- or) leaves bitwise logical or 32 bit	3-19
ORCASE	W siem	(--) extension of CASE; permits multiple selections	4-10

OUT	V fig	(-- addr) incremented by every EMIT	3-19
OVER	W stand	(d1/d2 -- d1/d2/d1) copies second stack value	3-19
P2P		(-- value) POINT TO POINT OPERATION MODE	4-64
PACK_FTH_DATA	W siem	(d1/d2/.../_of_d's -- message_buffer_addr -1)	5-3
PAD	C stand	(-- addr) leaves address of text output buffer	3-19
PAINT	W siem	(attr --) change active window to attribute	4-33
PAINT_FIELD	W siem	(row/col/length/attr --)	4-33
PAINT_KEY	W siem	(fk#/begin/end/color --)	4-40
PAINT_ROW	W siem	(row/attr --) paints specified row	4-33
PAR_ERR	W siem	(-- n) returns the attribute use by SEND_DATA	4-81
PARTITIONWD	W siem	(LS0/LS1/ -- -/LSn/#LS --) logically partitions Winchester Disk	4-28
PASS	C siem	(-- value) PRI channel mode pass through	4-76
PCM_FRAMING	C siem	(-- value) PRI pcm framing	4-75
PFA	W fig	(nfa -- pfa)	3-19
PH:ACTIVATE		(portid --) activates the S-bus	4-60
PH:DEACTIVATE		(portid --) deactivates the S-bus	4-60
PICK	W forth 83	(n -- copy_of_nth_stack_entry) n starts at 0	3-19
POINTER-M	V siem	(-- addr) display pointer for window M	4-37
POP_DATA	W siem	(--)	
POP_EVENT	W siem	(-- value) pop event from event queue onto stack	4-83
POP_ROW	W siem	(row --) pops designated row from window stack	4-35
POP_WINDOW	W siem	(wd --) equivalent to SHOW_WINDOW and OPEN_WINDOW	4-31
PORT1	C siem	(-- value) Test PORT1 (top group of connectors)	4-3
PORT1_A	C siem	(-- value) Test PORT1 transmitter	4-3
PORT1_ACBUF	C siem	(-- addr) addr of PORT1_A configuration buffer	4-54
PORT1_B	C siem	(-- value) Test PORT1 receiver	4-3
PORT1_BCBUF	C siem	(-- addr) addr of PORT1_B configuration buffer	4-54
PORT2	C siem	(-- value) Test PORT2 (Multi-port machines only)	4-3
PORT2_A	C siem	(-- value) Test PORT2 transmitter	4-3
PORT2_ACBUF	C siem	(-- addr) addr of PORT2_A configuration buffer	4-54
PORT2_B	C siem	(-- value) Test PORT2 receiver	4-3
PORT2_BCBUF	C siem	(-- addr) addr of PORT2_B configuration buffer	4-54
PORT3	C siem	(-- value) Test PORT3 (top group of connectors)	4-3
PORT3_A	C siem	(-- value) Test PORT3 transmitter	4-3
PORT3_ACBUF	C siem	(-- addr) addr of PORT3_A configuration buffer	4-54
PORT3_B	C siem	(-- value) Test PORT3 receiver	4-3
PORT3_BCBUF	C siem	(-- addr) addr of PORT3_B configuration buffer	4-54
PORT4	C siem	(-- value) Test PORT4 (Multi-port machines only)	4-3
PORT4_A	C siem	(-- value) Test PORT4 transmitter	4-3
PORT4_ACBUF	C siem	(-- addr) addr of PORT4_A configuration buffer	4-54
PORT4_B	C siem	(-- value) Test PORT4 receiver	4-3
PORT4_BCBUF	C siem	(-- addr) addr of PORT4_B configuration buffer	4-54
PORT_BRID	C siem	(-- value) BAI Port A (NT) and Port B (TE),	4-64
PORT_BRIDA	C siem	(-- value) BAI Port A (NT)	4-64
PORT_BRIDB	C siem	(-- value) BAI Port B (TE),	4-64
POWER_FEEDING	W siem	(portid / mode --) specifies, which power feeding source should be used	4-59
PPA	C siem	(-- value) PPA function number	4-89

PR_A/H_TABLE	W siem	(addr --) Transfers the address of the ASCII/HEX table for TRA_PRINT_MODE to the system.	4-52
PREFDOT	V siem	(-- addr) controls prefix for the .	3-19
PREVIOUS	W siem	(-- count/addr) previous frame in capture buffer	4-45
PRI_ALARM_GEN	W siem	(pri_pid / alarm_type / value --) switches the alarm generation	4-73
PRI_B2NORM	C siem	(-- value) PPI back to normal	4-76
PRI_BLUE_ALARM	C siem	(-- value) PRI blue alarm	4-76
PRI_BPVL	C siem	(-- value) PRI bipolar violation	4-76
PRI_CERR	C siem	(-- value) PRI cept special error	4-76
PRI_CRCR	C siem	(-- value) PRI CRC error	4-76
PRI_CHNL_MODE	W siem	(pri_pid / channel / mode --) Sets the selected channel to the specified mode	4-69
PRI_EOV	C siem	(-- value) PRI ERROR overflow	4-76
PRI_ERR_GEN	W siem	(pri_pid / err_rate / tot_rate / err_type --) Configures the selected PRI port to generate errors	4-75
PRI_ERR_IND	W siem	(pri_pid / err_type / interval / limit --) Enables an error report after INTERVAL * 100 multiframes received	4-75
PRI_ERR_STA	W siem	(pri_pid -- num_of_bpvl / num_of_crcr / num_of_ferr / num_of_merr) Reports a statistic of total errors detected by the specified PRI port	4-72
PRI_FERR	C siem	(-- value) pri frame error	4-76
PRI_FRAMING	C siem	(pri_pid/value --) Configures the PRI for the specified primary rate framing format	4-66
PRI_FRMERR_LOC	W siem	(pri_pid / value) Defines the framing error generation in bit 3 of EVEN frames or in bit 2 of ODD frames	4-68
PRI_I/F_MODE	W siem	(pri_pid/mode --) Establishes the interface mode for the specified port	4-66
PRI_IDLE_CHAR	W siem	(7 pri_pid/char --) Defines the idle character to be transmitted	4-67
PRI_IHDL	W siem	(pri_pid / Richtung / Zeitschlitz / value --) Enables or disables inverted HDLC reception or transmission for the selected port and channel	4-69
PRI_IMPEDANCE	C siem	(pri_pid/value --) Sets the primary rate source and termination impedances to the specified value	4-67
PRI_IND	C siem	(-- value) PRI INDICATION	4-76
PRI_LINE_CODE	W siem	(pri_pid / value --)	4-67
PRI_LINE_LENGTH	C siem	(pri_pid/value --) Supports transmit line length selection for two different distances	4-67
PRI_LMA	C siem	(-- value) PRI lost of multiframe alignment	4-76
PRI_LSL	C siem	(-- value) PRI lost of signal level	4-76
PRI_MERR	C siem	(-- value) PRI multiframe errors generated	4-76
PRI_MON	C siem	(-- value) PRI MONITOR MODE	4-75
PRI_NORMAL	C siem	(-- value) PRI channel mode normal	4-76
PRI_PLLNL	C siem	(-- value) PRI physical link not locked	4-76
PRI1_PORTA_CBUF	C siem	(-- value) PRI port a configuration buffer	4-75
PRI1_PORTA_CHB	C siem	(-- value) PRI port a channel ram table	4-75
PRI1_PORTB_CBUF	C siem	(-- value) PRI port b configuration buffer	4-75
PRI1_PORTB_CHB	C siem	(-- value) Ri PORT B channel ram table	4-75

PRI2_PORTA_CBUF	C siem	(-- value) PRI port a configuration buffer	4-75
PRI2_PORTA_CHB	C siem	(-- value) PRI port a channel ram table	4-75
PRI2_PORTB_CBUF	C siem	(-- value) PRI port b configuration buffer	4-75
PRI2_PORTB_CHB	C siem	(-- value) RI PORT B channel ram table	4-75
PRI_RED_ALARM	C siem	(-- value) PRI RED ALARM	4-76
PRI_RNAT_BITS	W siem	(pri_pid / nat_bits --) Extracts the National bits from timeslot 0 of odd frames	4-71
PRI_ROUTE	W siem	(channel / channel-type / pri_pid / port_d --) Routes the channel or list of channels to the FECp port	4-72
PRI_RSI_BIT	W siem	(pri_pid / si_ind / -- si_bit) Extracts the specified SI bit from bit 1 of TS0 in frame 15 and bit 1 TS0 in frame 13	4-70
PRI_TINT_BIT	W siem	(pri_pid / int_bit --) Defines the transmitted International bit in bit 1 of all TS0	4-70
PRI_TNAT_BITS	W siem	(pri_pid / nat_bits --)	4-70
PRI_TSI_BIT	W siem	(pri_pid / si_ind / value --) Sets / clears the SI bits to be transmitted	4-70
PRI_TX_EXT	C siem	(pri_pid/value --) Selects whether any 64kbps channel intransceiver A and transceiver B of the PRI may be accessed externally	4-67
PRI_YEL_ALARM	C siem	(-- value) PRI YELLOW ALARM	4-76
PRI1	C siem	(-- value) PRI port a	4-75
PRI1PTA	W siem	(-- value) PRI PORT A	4-75
PRI1PTB	W siem	(-- value) PRI PORT B	4-75
PRI2	C siem	(-- value) PRI port b	4-75
PRI2PTA	W siem	(-- value) PRI PORT A	4-75
PRI2PTB	W siem	(-- value) PRI PORT B	4-75
PRINT	W siem	(addr/count/byte_word_flag -- status)	4-45
PRINT_EOL	W siem	(--) Print end of line	4-46
PRINT_IND	C siem	(-- value) printer indication	4-83
PRINT_Q_EMP	W siem	(-- value) printer queue empty	4-84
PRINT_W	W siem	(addr/count/byte_word_flag --)	4-46
PRINTER	C siem	(-- value) specifies printer port for CONFIG	4-3
PRINTER_CBUF	W siem	(-- addr)	4-54
PRINTER_EOL	W siem	(id --)	4-46
PRINTER_MODE	W siem	(mode --)	4-53
PRIPTE	W siem	(-- value) PRI external access port	4-75
PROTOCOL	W siem	(port/type --) specify protocol	4-48
QUERY	W stand	(--) inputs 80 characters of text from terminal	3-19
QUICKY	C siem	(MODE ---) activates fast compiler	6-3
QUIT	W stand	(--) clears return stack, stops compilation	3-19
R	W stand	(-- d) copies top of return stack to comp. stack	3-19
R#	V siem	(-- addr)	3-19
R>	W stand	(-- d)	3-19
R0	V fig	(-- addr) initial address of return stack	3-19
RDIR	W siem	(filename/dir_type --addr_of_dir_buffer/ no_of_lines	4-26
RDISC	W siem	(-- value) receive channel disconnected	4-76
RE	C siem	(-- value) READ access type	4-21

RE_ROUTE	W siem	(function_no --) reroute message buffer	4-88
READ_BLOCKS	W siem	(file_desc / buffer_addr / no_of_blocks --)	4-23
REBOOT	W siem	(--) complete rebooting of system disk	4-6
RECEIVER	C siem	(-- value) PRI RECEIVER DIRECTION	4-76
RED_BG	C siem	(-- value) red background	4-36
RED_FG	C siem	(-- value) red foreground	4-36
REDO	C siem	(--) Performs a re-compilation starting at FILENAME.	6-2
REMOTE	C siem	(-- value) serial async. port 1	4-3
REMOTE_CBUF	W siem	(-- addr)	4-54
REMOTE_OUT	W siem	(addr / count / byte_word_flag -- status)	4-46
REMOTE_OUT_W	W siem	(addr / count / byte_word_flag --)	4-46
REMOVE	W siem	(filename --) removes file from directory	4-23
RENAME	W siem	(old_filename / new_filename --) renames file	4-24
REPEAT	W stand	(addr / d --) for BEGIN WHILE REPEAT loops	3-20
REPEAT_KEY_SEQ	W siem	(--) closes off user-defined key sequence	4-41
REPORT	W siem	(--) report status for last disk operation	4-26
REPT	C siem	(-- value) repeat argument for SEND_DATA	4-81
REST_IDLE	W siem	(port / value --) specify idle condition prior to transmission	4-50
RESTORE_DSPP	W siem	(--) restores cursor position and attribute	4-36
RESTORE_SCREEN	W siem	(--) returns saved screen image in returns it to display screen	4-33
RETURN_STATE	W siem	(--) Permits the user to program the equivalent of subroutine calls into the state machine.	8-2
REV_VIDEO	C siem	(-- value) reverse	4-36
REWR	C siem	(-- value) READ/WRITE access type video	4-21
RMT_BLK_MODE	W siem	(--)	4-53
RMT_CHR_MODE	W siem	(fifo_addr / size_of_fifo --)	4-53
RMT_CD	C siem	(--) carrier detect	4-84
RMT_CS	C siem	(--) clear to send	4-84
RMT_DTR	C siem	(-- value) remote DTR	4-46
RMTGETC	W siem	(-- character [-1]) gets character from remote	4-84
RMTGETS	W siem	(-- addr / count) gets string from remote	4-84
RMT_INTERPRET	W siem	(--) remote interpret	4-84
RMT_LEADS	C siem	(-- value) remote lead transition	4-83
RMT_Q_EMP	C siem	(-- value) remote queue empty	4-83
RMT_RS	C siem	(-- value) remote RTS	4-46
RMT_STR_IND	C siem	(-- value) remote console string indication	4-83
ROT	W stand	(d1 / d2 / d3 -- d2 / d3 / d1)	3-20
Rows	W siem	(-- rows) returns number of rows in window	4-35
Rowtable	W siem	(-- addr) returns address of row table	4-35
RP!	W fig	(--) initialize return stack pointer	3-20
RP@	W stand	(--) return stack address	3-20
RPT_D_ERR	W siem	(status --) report disk error/status	4-25
RST_INQ	W siem	(--) reset terminal input queue	4-35
RTEEND	W siem	(-- addr) returns address of first byte past end of local's system code	4-9
RTEEND_M	W siem	(-- addr) returns address of first byte past end of main's system code	4-9

RTEEND_1	W siem	(-- addr) returns address of first byte past end of CPU1 system code	4-9
RTEEND_2	W siem	(-- addr) returns address of first byte past end of CPU2 system code	4-9
RTEEND_3	W siem	(-- addr) returns address of first byte past end of CPU3 system code	4-9
RTEEND_4	W siem	(-- addr) returns address of first byte past end of CPU4 system code	4-9
RTEEND_5	W siem	(-- addr) returns address of first byte past end of CPU5 system code	4-9
RTEEND_6	W siem	(-- addr) returns address of first byte past end of CPU6 system code	4-9
RTEEND_7	W siem	(-- addr) returns address of first byte past end of CPU7 system code	4-9
RTEEND_8	W siem	(-- addr) returns address of first byte past end of CPU8 system code	4-9
RUN_PROGRAM	W siem	(filename --) runs program whose name is on stack	4-6
RUN_RTE		(--) runs other functions in the system	4-7
RUN_WORD	W siem-sys	(string_addr --) search and execute word	4-7
RX_CH1	W siem	(-- TO_DCE rx id port 1/TO_DTE rx id port 1)	4-86
RX_CH2	W siem	(-- TO_DCE rx id port 2/TO_DTE rx id port 2)	4-86
RX_CH3	W siem	(-- TO_DCE rx id port 3/TO_DTE rx id port 3)	4-86
RX_CH4	W siem	(-- TO_DCE rx id port 4/TO_DTE rx id port 4)	4-86
RX_OFF	W siem	(port --) stop frame indications	4-77
RX_ON	W siem	(port --) start frame indications	4-77
RX_SPEED	W siem	(port/value --) sets baud rate for receiver	4-52
S	W siem	(-- n) copies element from top of shell stack & pushes onto parameter stack	4-44
S->N	W stand	(startaddr -- 0 value / 1) N->S does ASCII string conversion to a number.	3-20
S>	W siem	(-- n) pops item from shell stack and pushes parameter stack	4-44
S.	W stand	(d -- d) like '.', but non-destructive	3-20
S0	V stand	(-- addr) initial address of parameter stack	3-20
SAVE_ATTR	W siem	(-- addr) saves attribute	4-36
SAVE_DSPP	W siem	(--) saves display pointer	4-36
SAVE_SCREEN	W siem	(--) Save an image of the current screen's window positions	4-33
SAVEB	W siem	(filename --) Saves the present user words	4-5
SC@	W siem	(addr -- signed_b)	3-20
SEARCH	W siem	(string ---) display all Forth-Definitions, which contain 'string' in their name-field.	6-3
SEEK	W siem	(file desc/offset --) seeks into the file	4-23
SELL_WINDOW	W siem	(window_descriptor --) sells designated window	4-30
SEND_DATA	W siem	(port/buffer/attr --) transmits data	4-81
SEND_DATA_WAIT	W siem	(port/buffer/attr --) data sent to send/rec func.	4-81
SET_ATTR	W siem	(attr --) set attribute	4-36
SET_CUR_DEV	W siem	(device_number --) sets current selected drives	4-21
SET_PRIORITY	W siem	(portid / priority --) specifies the access priority for D channel on the S-Bus	4-58

SET_RETURN	W siem-sys (--) set return	4-43
SET_SCREEN_POINTER	W siem-sys (--) sets POINTER-M using THIS-RX array pointer	4-37
SET_TIME	W siem (YY/MO/DD/HH/MI/SS --) set real-time clock	4-19
SF	C siem (-- value) shifted function key group	4-42
SF1	C siem (-- value) shifted function key 1	4-91
SF2	C siem (-- value) shifted function key 2	4-91
SF3	C siem (-- value) shifted function key 3	4-91
SF4	C siem (-- value) shifted function key 4	4-91
SF5	C siem (-- value) shifted function key 5	4-91
SF6	C siem (-- value) shifted function key 6	4-91
SF7	C siem (-- value) shifted function key 7	4-91
SF8	C siem (-- value) shifted function key 8	4-91
SHELL{	W siem (--) initiates protective shell for word following	4-43
SHIFT_WINDOW	W siem (n --) shift open window vertically on screen	4-31
SHORT_BUS	C siem (-- value) short bus operation mode,	4-64
SHOW_WINDOW	W siem (wd --) makes indicated window visible on screen	4-31
SHSP!	W siem (--) resets shell stack pointer to point to bottom of shell stack	4-44
SI1	C siem (-- value) Parameter for PRI_TS1_Bit	4-76
SI2	C siem (-- value) Parameter for PRI_TS2_Bit	4-76
SIGN	W stand (n/d -- d) sign for pictured number output	3-20
SLEEP	W siem (#_of_one_tenth_of_a_second --)	4-18
SMUDGE	W fig (--) toggle smudge bit in last definition	3-20
SORT_ON	W siem (yes / no --) switches the time stamp sort algorithm ON or OFF. The default value is Yes.	4-20
SP!	W fig (resets) initialize stack pointer from S0	3-20
SP@	W stand (-- addr) return addr. of stack position to top	3-20
SPACE	W stand (--) transmits an ASCII blank to the screen	3-20
SPACES	W stand (n --) Transmits n ASCII blanks to the screen	3-20
SPC	W siem (-- value) send steady space	4-81
SPEED	W siem (port/value --) bit per second rate	4-49
SPEED_TABLE	V siem-sys (-- addr) returns address of speed table	4-49
STACK	W fig (--) prints stack	3-21
START_TIMER	W siem (timer_no_/time --)	4-18
START_TIMER_20	W siem (timer_no/time --)	4-18
STATE	V stand (-- addr) interpreting state, non-zero=compiling	3-21
STATE?	W siem (portid -- / -- state) returns the state of Layer1 device due to the interface mode	4-60
STATE{	W siem (Zustandsnummer --) Start of STATE	8-2
Status	W siem (-- addr) returns address of status word	4-35
STATUS-W	C siem (-- wd)	4-34
STD_CLK	W siem (port --) standard clocking	4-49
STOP_BITS	W siem (port/value --) number of stop bits	4-50
STOP_TIMER	W siem (timer_no --)	4-18
STR_IND	C siem (-- value) string indication	4-83
STRIP_SYNC	W siem (port/value --) specify string sync control	4-51
STYPE	W siem (start_addr/no_of_char --) fast version of TYPE	4-36
STYPE_A	W siem (start_addr/no_of_char/attr --) STYPE with attr.	4-36
SW@	W siem (addr -- signed_n)	3-21

SWAP	W stand	(d1/d2 -- d2/d1)	3-21
SWAPB	W siem	(b1/b2 -- b2/b1)	3-21
SWITCH	W siem	(FORTH# --) connects to designated FORTH machine	5-2
SWITCH_IN2OUT	W siem	(portid / incoming data stream / outgoing data stream --) connects the incoming data stream	4-62
SWITCHED		(-- value) FORTH machine just been switched to	4-83
SYNC_CHAR	W siem	(port/value --) specify sync character	4-50
SYNC_RESET	W siem	(port/value --) specify sync reset character	4-50
SYSERR_IND	C siem	(-- value) system error indication	4-83
TAKE_ERROR_RETURN	W siem-sys	(-- 0) take error return	4-43
TAKE_RETURN	W siem-sys	(-- 1) take normal return	4-43
TBASEYR	W siem	(-- base year) returns value of base year	4-19
TCF	C siem	(-- value) terminal/console function number	4-89
TCLR	W siem	(--) initializes the state machine data structure.	8-3
TE	C siem	(-- value) TERMINAL EQUIPMENT	4-64
TERM_RES	W siem	(portid / mode --) specifies, whether the termination resistor on the S0 connector module should be switched to 100 Ohm or to High Impedance	4-59
THEN	W stand	(addr/d --) for IF ELSE THEN	3-21
THERE	W siem	(row/col --) positions cursor to row/col	4-32
THIS-RX	W siem-sys	(-- addr) used by SET_SCREEN_POINTER	4-37
TIB	V fig	(-- addr) pointer to terminal input buffer	3-21
TIMER_IND	C siem	(-- value) timer timeout indication	4-83
TM_RUN	W siem	(--) starts the state machine running in state 0.	8-3
TM_STOP	W siem	(--) is used to terminate the operation of the Test Manager state machine.	8-3
TO		(--) Defines the cell aaaa in the dictionary word nnnn to be replaced by bbbb.	6-2
TODCE	C siem	(-- value) emulates a DCE	4-47
TODTE	C siem	(-- value) emulates a terminal	4-47
TOGGLE	W fig	(addr/b --) complements contents of addr	3-21
TOGGLEW	W siem	(addr/n --) same with 16 bit mask	3-21
TOPE	W siem	(tone --) sets the tone of beeper	4-18
TRANSMITTER	W siem	(-- value) PRI-Senderrichtung	4-76
TRAVERSE	W fig	(addr1/n -- addr2) traverse variable length name field	
TRUNCATE	C siem	(-- d) truncate parameter	4-22
TS0_REG	W siem	(-- value) PRI TS0 REGISTER	4-76
TSTAMP_ADD	W siem	(addr_ts_1/addr_ts_2/addr_for_sum --)	4-19
TSTAMP_COMP	W siem	(timestamp_1_addr/timestamp_2_addr -- =?)	
		< -1 = 0 > +1	4-19
TSTAMP_SUB	W siem	(addr_ts_1/addr_ts_2/addr_for_diff -- flag)	
		subtracts 2 48 bit values. The flag is -1 if second time is greater than the first.	4-19
TTX	C siem	(-- value) TELETEX character set	4-36
TURN_OFF	W siem	(port/lead --) turns off interface lead	4-13
TURN_ON	W siem	(port/lead --) turns on interface lead	4-13
TX_CONTROL	W siem	(port/id --) sets TRANSMITTER control mode	4-52
TX_SPEED	W siem	(port/value --) sets baud rate for transmitter]	4-52

TYPE	W stand	(addr/count --) transmits count char. from addr.	3-21
TYPH	W siem	(addr/count --) TYPE version for HEX output	3-21
U*	W stand	(u1/u2 -- ud) unsigned multiply	3-21
U/	W fig	(ud/u1 -- u2/u3) like/MOD, but unsigned	3-21
UF	C siem	(-- value) unshifted function key group	4-42
UF1	C siem	(-- value) unshifted function key 1	4-91
UF2	C siem	(-- value) unshifted function key 2	4-91
UF3	C siem	(-- value) unshifted function key 3	4-91
UF4	C siem	(-- value) unshifted function key 4	4-91
UF5	C siem	(-- value) unshifted function key 5	4-91
UF6	C siem	(-- value) unshifted function key 6	4-91
UF7	C siem	(-- value) unshifted function key 7	4-91
UF8	C siem	(-- value) unshifted function key 8	4-91
UN_EMULATION	C siem	(-- value) Parameter for PRI_I/F_MODE: User/Network-Mode	4-75
UNALTER	W siem	(--) Undefined all the changes to the dictionary word nnnn.	6-2
UNLITE_KEY	W siem	(fk# --)	4-39
UNLOCK_WINDOW	W siem	(window --) reverses effect of LOCK_WINDOW	4-34
UNPACK_DATA	W siem	(data_addr -- status/data/ BSC/end_timestamp/start_timestamp/port)	4-86
UNPACK_PRI	W siem	(address -- pri_event_type / portid / timestamp_address / blockseqcount / error_count / cept_state) unpacks a received PRI indication buffer.	4-75
UNTIL	W stand	for BEGIN UNTIL loops	3-21
USART1 to USART8	C siem	(-- value) PORT6_B function number	4-89
UU_EMULATION	C siem	(-- value) Parameter for PRI_IF_MODE: User- User	4-75
VARIABLE	W stand	(d --) define variable: initial -- varname	3-22
VLIST	W stand	(--) list dictionary from latest to first entry	3-22
VOC-LINK	V fig	(-- addr) points to last vocabulary definition	3-22
VOCABULARY	W stand	(--) start vocabulary definition	3-22
VOCIN	W siem-sys	(--) vocabulary initialization for ABORT	3-22
VRC	C siem	(-- value) VRC/LCR BCC type	4-51
W<	W siem	(n1/n2 -- f) like '<', but for 16 bit comparison	3-22
W.	W siem	(d --)	4-32
W."	W siem	(--) output string sent to active window	4-32
W.H	W siem	(d --)	4-32
W.MOVE	W siem	(wstring/wcount --) outputs string of words	4-32
W.R	W siem	(n / columns --) (-- ascii_value)	4-33
W.SPACES	W siem	(n --) display n Spaces on the window	4-33
W.TYPE	W siem	(string/count --)	4-32
W.TYPE_A	W siem	(string/count/attr --)	4-32
W.TYPE_RC	W siem	(string/count/row/col --)	4-32
W!	W siem	(n/Adr)	3-22
W,	W siem	(n --) like ',', but for 16 bit	3-22
W@	W siem	(addr -- n) 16 bit fetch	3-22
W-ADDRESS	W siem	(-- addr) returns beginning RAM addr. of window	4-34
W=	W siem	(n1/n2 -- f) like '=', but for 16 bit comparison	3-22

W>	W siem	(n1/n2 -- f) like '>', but for 16 bit comparison	3-22
WAIT_EVENT	W siem	(--) wait for event	4-83
Wcb	W siem-sys	(-- addr)	4-34
Wcb_table	W siem-sys	(-- addr)	4-34
WCMOVE	W siem	(src/dst/length --) pack words to bytes	3-22
WCR	W siem	(--) performs carriage return on active window	4-33
WD#	W siem	(-- window_descriptor) returns window descriptor	4-35
WD0 bis WD7	W siem	(--)	4-27
WD10 bis WD17	W siem	(--)	4-27
WDOWN	W siem	(--) sets output direction of window to down	4-31
WHERE	W siem	(-- row/col) returns cursor of active window	4-32
WHILE	W stand	(f --) for BEGIN WHILE REPEAT loops	3-23
WHI_BG	C siem	(-- value) white background	4-36
WHI_FG	C siem	(-- value) white foreground	4-36
WIDTH	V fig	(-- addr) width of name field	3-23
Window#	V siem	(-- addr) returns addr of active window stack	4-34
WORD	W stand	(c --) parse one word from input to HERE	3-23
WR	C siem	(-- value) write access type	4-21
WRAP	W siem	(--) makes selected window a wraparound window	4-31
WRITE_BLOCKS	W siem	(file_desc/buffer_addr/no_of_blocks --)	4-23
WRTprt_FS	W siem	({YES NO} --) sets software write protect flag	4-24
WSCROLL	W siem	(--) makes selected window a scrolling window	4-31
WSTACK	W siem	(-- addr) returns address of window stack	4-35
WUP	W siem	(--) sets output direction of window to up	4-31
WWR	C siem	(-- value) write with wraparound access type	4-21
X.21	C siem	(-- value)	4-48
X"	W siem	(-- addr) HEX string input	3-23
XOR	W stand	(d1/d2 -- xor) leaves bitwise logical exclusive	3-23
YEL_BG	C siem	(-- value) yellow background	4-36
YEL_FG	C siem	(-- value) yellow foreground	4-36
YES	C siem	(-- 1) used when true flag is desired	3-23

Miscellaneous

for Protocol Tester K1197

SIEMENS AG

Version : January 1990

Table of Contents

1	Character sets	1-1
2	Bibliography	2-1
3	Electromagnetic Compability	3-1
3.1	Notes	3-1
3.2	Manufacturer's Certificate	3-1

1 Character sets

B7 B6 BITS B4 B3 B2 B1				ROW	0 0 0		0 0 1		0 1 0		0 1 1		1 0 0		1 0 1		1 1 0		1 1 1	
					COLUMN 0		1		2		3		4		5		6		7	
B4	B3	B2	B1	ROW																
0	0	0	0	0	NUL SPAC	0 0 0	DLE P	20 16 10	SP	40 21 20	0	60 48 30	@	100 64 40	P	120 80 50	'	140 96 60	p	160 112 70
0	0	0	1	1	SOH A	1 1 1	DC1 (XON) O	21 17 11	!	41 33 21	1	61 49 31	A	101 65 41	Q	121 81 51	a	141 97 61	q	161 113 71
0	0	1	0	2	STX B	2 2 2	DC2 R	22 18 12	"	42 34 22	2	62 50 32	B	102 66 42	R	122 82 52	b	142 98 62	r	162 114 72
0	0	1	1	3	ETX C	3 3 3	DC3 (XOFF)	23 19 13	#	43 35 23	3	63 51 33	C	103 67 43	S	123 83 53	c	143 99 63	s	163 115 73
0	1	0	0	4	EOT D	4 4 4	DC4 T	24 20 14	\$	44 36 24	4	64 52 34	D	104 68 44	T	124 84 54	d	144 100 64	t	164 116 74
0	1	0	1	5	ENQ E	5 5 5	NAK U	25 21 15	%	45 37 25	5	65 53 35	E	105 69 45	U	125 85 55	e	145 101 65	u	165 117 75
0	1	1	0	6	ACK F	6 6 6	SYN V	26 22 16	&	46 38 26	6	66 54 36	F	106 70 46	V	126 86 56	f	146 102 66	v	166 118 76
0	1	1	1	7	BEL G	7 7 7	ETB W	27 23 17	'	47 39 27	7	67 55 37	G	107 71 47	W	127 87 57	g	147 103 67	w	167 119 77
1	0	0	0	8	BS H	8 8 8	CAN X	30 24 18	(	50 40 28	8	70 56 38	H	110 72 48	X	130 88 58	h	150 104 68	x	170 120 78
1	0	0	1	9	HT I	11 9 9	EM Y	31 25 19	)	51 41 29	9	71 57 39	I	111 73 49	Y	131 89 59	i	151 105 69	y	171 121 79
1	0	1	0	10	LF J	12 10 A	SUB Z	32 26 1A	.	52 42 2A	:	72 58 3A	J	112 74 4A	Z	132 90 60	j	152 106 6A	z	172 122 7A
1	0	1	1	11	VT K	13 11 B	ESC [	33 27 1B	+	53 43 2B	;	73 59 3B	K	113 75 4B	[	133 91 61	k	153 107 6B	{	173 123 7B
1	1	0	0	12	FF L	14 12 C	FS \ 	34 28 1C	,	54 44 2C	<	74 60 3C	L	114 76 4C	\	134 92 62	l	154 108 6C		174 124 7C
1	1	0	1	13	CR M	15 13 D	GS)	35 29 1D	-	55 45 2D	=	75 61 3D	M	115 77 4D	]	135 93 63	m	155 109 6D	}	175 125 7D
1	1	1	0	14	SO N	16 14 E	RS ~	36 30 1E	.	56 46 2E	>	76 62 3E	N	116 78 4E	~	136 94 64	n	156 110 6E	~	176 126 7E
1	1	1	1	15	SI O	17 15 F	US ?	37 31 1F	/	57 47 2F	?	77 63 3F	O	117 79 4F	-	137 95 65	o	157 111 6F	DEL	177 127 7F

KEY



ASCII-Character set

Bit positions

								0				0				1				1							
								0				1				0				1							
								0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1				
								0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1				
								Spalte																			
								0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F				
b ₈	b ₇	b ₆	b ₅	b ₄	b ₃	b ₂	b ₁	Zeile																			
0	0	0	0	0	0	0	0	NUL	DLE	DS			SP	&	-					{	}	\	0				
0	0	0	0	1	1	1	1	SOH	DC1	SOS				/			a	j	~		A	J	1				
0	0	1	0	0	0	0	0	STX	DC2	FS	SYN						b	k	s		B	K	S 2				
0	0	1	1	1	1	1	1	ETX	TM								c	i	t		C	L	T 3				
0	1	0	0	0	0	0	0	PF	RES	BYP	PN						d	m	u		D	M	U 4				
0	1	0	1	1	1	1	1	HT	NL	LF	RS						e	n	v		E	N	V 5				
0	1	1	0	0	0	0	0	LC	BS	ETB	UC						f	o	w		F	O	W 6				
0	1	1	1	1	1	1	1	DEL	IL	ESC	EOT						g	p	x		G	P	X 7				
1	0	0	0	0	0	0	0		CAN								h	q	y		H	Q	Y 8				
1	0	0	1	1	1	1	1	RLF	EM								\	i	r	z		I	R	Z 9			
1	0	1	0	0	0	0	0	A	SMM	CC	SM			ø	'	:	:										
1	1	0	1	1	1	1	1	B	VT					•	\$	.	#										
1	1	0	0	0	0	0	0	C	FF	IFS		DC4	<	"	%	@											
1	1	0	1	1	1	1	1	D	CR	IGS	ENQ	NAK	(	)	_	'											
1	1	1	1	0	0	0	0	E	SO	IRS	ACK		+	:	>	=											
1	1	1	1	1	1	1	1	F	SI	IUS	BEL	SUB	j	1	?	"											

EBCDIC-Character set

				8	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
				7	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
				6	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
				5	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
					02	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
4	3	2	1																	
0	0	0	0	00			SP	0	@	P		p				*			Ω	k
0	0	0	1	01			!	1	A	Q	a	q			i	±	'		Æ	α
0	0	1	0	02			-	2	B	R	b	r			e	2	'		⊠	đ
0	0	1	1	03				3	C	S	c	s			£	3	^		Ⓐ	đ
0	1	0	0	04				4	D	T	d	t			\$	x	~		Ⓓ	ñ
0	1	0	1	05			%	5	E	U	e	u			¥	μ	-			ı
0	1	1	0	06			&	6	F	V	f	v			#	Π			ıı	ıı
0	1	1	1	07			'	7	G	W	g	w			§	•	•		Ł	ł
1	0	0	0	08			(	8	H	X	h	x			⌘	+	••		Ł	ł
1	0	0	1	09			)	9	I	Y	i	y							Ø	ø
1	0	1	0	10			*		J	Z	j	z					•		Œ	œ
1	0	1	1	11			+	:	K	[	k				«	»	.		⓪	β
1	1	0	0	12			.	<	L		l					1/4			Ⓟ	Ⓟ
1	1	0	1	13			-	=	M	]	m					1/2	"		Ⓕ	Ⓕ
1	1	1	0	14			.	>	N		n					3/4	'		Ⓠ	Ⓠ
1	1	1	1	15			/	?	O		o					ı	˘		ˆ	

TELETEX-Character set

TABLE 1/T.61
The Teletex primary set of grafic characters

Position	Graphic	Name or description	Position	Graphic	Name or description	Position	Graphic	Name or description
2/1	!	exclamation mark	4/0	@	commercial at	6/0		(not used)
2/2	"	quotation mark	4/1	A	capital A	6/1	a	small a
2/3		(not used)	4/2	B	capital B	6/2	b	small b
2/4		(not used)	4/3	C	capital C	6/3	c	small c
2/5	%	percent sign	4/4	D	capital D	6/4	d	small d
2/6	&	ampersand	4/5	E	capital E	6/5	e	small e
2/7	'	apostrophe	4/6	F	capital F	6/6	f	small f
2/8	(	left parenthesis	4/7	G	capital G	6/7	g	small g
2/9	)	right parenthesis	4/8	H	capital H	6/8	h	small h
2/10	*	asterisk	4/9	I	capital I	6/9	i	small i
2/11	+	plus sign	4/10	J	capital J	6/10	j	small j
2/12	,	comma	4/11	K	capital K	6/11	k	small k
2/13	-	hyphen or minus sign	4/12	L	capital L	6/12	l	small l
2/14	.	full stop, period	4/13	M	capital M	6/13	m	small m
2/15	/	solidus	4/14	N	capital N	6/14	n	small n
3/0	0	digit 0	4/15	O	capital O	6/15	o	small o
3/1	1	digit 1	5/0	P	capital P	7/0	p	small p
3/2	2	digit 2	5/1	Q	capital Q	7/1	q	small q
3/3	3	digit 3	5/2	R	capital R	7/2	r	small r
3/4	4	digit 4	5/3	S	capital S	7/3	s	small s
3/5	5	digit 5	5/4	T	capital T	7/4	t	small t
3/6	6	digit 6	5/5	U	capital U	7/5	u	small u
3/7	7	digit 7	5/6	V	capital V	7/6	v	small v
3/8	8	digit 8	5/7	W	capital W	7/7	w	small w
3/9	9	digit 9	5/8	X	capital X	7/8	x	small x
3/10	:	colon	5/9	Y	capital Y	7/9	y	small y
3/11	;	semicolon	5/10	Z	capital Z	7/10	z	small z
3/12	<	less-than sign	5/11	[	left square bracket	7/11		(not used)
3/13	=	equals sign	5/12		(not used)	7/12		vertical line
3/14	>	greater-than sign	5/13	]	right square bracket	7/13		(not used)
3/15	?	question mark	5/14		(not used)	7/14		(not used)
			5/15 ^{a)}	-	low line	7/15		

^{a)} When interworking with Videotex, this code shall have the meaning *delimiter*.

TABLE 2/T.61
The Teletex supplementary set of graphic characters

Position	Graphic	Name or description	Position	Graphic	Name or description	Position	Graphic	Name or description
10/1	!	inverted exclamation mark	12/0	□	(not used)	14/0	Œ	ohm sign
10/2	¢	cent sign	12/1	□	grave accent	14/1	Æ	capital Æ diphthong
10/3	£	pound sign	12/2	□	acute accent	14/2	Ð	capital D with stroke
10/4	\$	dollar sign	12/3	□	circumflex accent	14/3	ª	ordinal indicator, feminine
10/5	¥	yen sign	12/4	□	tilde	14/4	Œ	capital H with stroke
10/6	¢	number sign	12/5	□	macron	14/5	(not used)	(not used)
10/7	§	section sign	12/6	□	breve	14/6	Œ	capital J ligature
10/8	¤	currency symbol	12/7	□	dot	14/7	Œ	capital L with middle dot
10/9	¤	(not used)	12/8	□	diaeresis or umlaut mark	14/8	Œ	capital L with stroke
10/10	(not used)	(not used)	12/9b)	□	(not used)	14/9	Œ	capital O with slash
10/11	°	angle quotation mark left	12/10	□	ring	14/10	Œ	capital Œ ligature
10/12	(not used)	(not used)	12/11	□	cedilla	14/11	Œ	ordinal indicator, masculine
10/13	(not used)	(not used)	12/12	□	non-spacing underline	14/12	Þ	capital thorn, Icelandic
10/14	(not used)	(not used)	12/13	□	double acute accent	14/13	Œ	capital T with stroke
10/15	(not used)	(not used)	12/14	□	ogonek	14/14	Œ	capital eng, Lapp
11/0	•	degree sign	12/15	□	caron	14/15	ˆ	small n with apostrophe
11/1	±	plus/minus sign	13/0	(not used)	(not used)	15/0	ƙ	small k, Greenlandic
11/2	²	superscript 2	13/1	(not used)	(not used)	15/1	æ	small æ diphthong
11/3	³	superscript 3	13/2	(not used)	(not used)	15/2	đ	small d with stroke
11/4	×	multiply sign	13/3	(not used)	(not used)	15/3	ð	small eth, Icelandic
11/5	µ	micro sign	13/4	(not used)	(not used)	15/4	h	small h with stroke
11/6	¶	paragraph sign, pilcrow	13/5	(not used)	(not used)	15/5	ı	small i without dot
11/7	·	middle dot	13/6	(not used)	(not used)	15/6	ı	small ij ligature
11/8	÷	divide sign	13/7	(not used)	(not used)	15/7	ı	small l with middle dot
11/9	(not used)	(not used)	13/8	(not used)	(not used)	15/8	ı	small l with stroke
11/10	(not used)	(not used)	13/9	(not used)	(not used)	15/9	ø	small o with slash
11/11	°	angle quotation mark right	13/10	(not used)	(not used)	15/10	œ	small œ ligature
11/12	¼	fraction one quarter	13/11	(not used)	(not used)	15/11	ß	small sharp s, German
11/13	½	fraction one half	13/12	(not used)	(not used)	15/12	þ	small thorn, Icelandic
11/14	¾	fraction three quarters	13/13	(not used)	(not used)	15/13	ı	small t with stroke
11/15	¿	inverted question mark left	12/14	(not used)	(not used)	15/14	Œ	small eng, Lapp
			12/15	(not used)	(not used)			

a) Diacritical marks are illustrated together with a rectangle representation the relative position of the graphic characters with which they are normally associated

b) In the 1980 version of this Recommendation code 12/9 was allocated to represent the umlaut mark. The use of this facility is discouraged. Its removal is foreseen in the future

2 Bibliography

- | | | |
|---|---------------------------------|---|
| THE COMPLETE FORTH | Alan Winfield | SIGMA Technical Press, US,
or John Wiley & Sons, Inc. ISBN |
| - very good, compact introduction; lots of comparisons with BASIC. | | |
| STARTING FORTH | Leo Brodie | Prentice-Hall, Inc. Eaglewood Cliffs,
NJ 07632
ISBN 0-13-842922-7 |
| - very good introduction, even total newcomers, cover remarkable depth. Refers to FORTH of FORTH, Inc. | | |
| PROGRAMMIEREN IN FORTH | Leo Brodie | Prentice-Hall, Inc. Hanser
ISBN 3-446-14070-0 |
| - sehr gute Einführung, auch für Anfänger; mit hervorragender Tiefe. Bezieht sich auf "polyFORTH". | | |
| MASTERING FORTH | Anita Anderson
Martin Trascy | Brady communications Comp. Inc.
Bowie, Maryland 20715
ISBN 0-89303-660-9 |
| - similar to starting FORTH, but starting at a higher level, with very good tools as examples. Refers to FORTH 83/Micromotion | | |
| FORTH ENCYCLOPEDIA | Mitch Derick
Linda Baker | Mountain View Press, Inc. PO Box
4656 Mountain View, CA 94040
ISBN - None - |
| - if you really want to know how the commands work. Refer to 8080 Version 1.1 fig. Without introduction to FORTH. | | |
| SYSTEMS GUIDE TO
FIG_FORTH | C. H. Ting | Offete Enterprises, Inc. San Mateo
CA 94040 ISBN |
| - similar to previous, + Syntax Backus Normal Form, EDITOR, PDP11, 8080-Assembler. | | |
| THINKING FORTH | Leo Brodie | Prentice-Hall, Inc. Eaglewood Cliffs,
NJ 07632
ISBN 0-13-917568-7 |
| - A very good introduction to FORTH design methodology. | | |

FORTH DIMENSIONS

Newsletter of FORTH

Interest Group ISBN

FORTH

W.P. Salam
O. Tisserand
B. TouloutEdition Francais
Eyrolles
61, Boulevard Saint-Germain,
75240 Paris CEDEX 05

- Tres comprehensible, avec des beaux exemples..

DER EINSTIEG IN FORTH

Paul M. Chilian

Markt & Technik Verlag
ISBN 3-89090-085-2

- Die Programmiersprache im Selbststudium erlernen. Mit zahlreichen Übungsaufgaben.

POCKET GUIDE "FORTH"

Pitman Publishing Ltd.
ISBN 0-273-02108-7

- Programming pocket guide.

ALL ABOUT FORTH

Mountain View Press, Inc.
ISBN 0-914699-00-8

- An annotated glossary. MVP-FORTH Series Volume 1 (second edition).

DIE PROGRAMMIERSPRACHE
FORTH

Ronald Zech

Franzis Verlag
ISBN 3-7723-7262-7

- Eine kompakte Darstellung der Programmiersprache mit vielen Beispielen.

3 Electromagnetic Compatibility

3.1 Notes

The protocol tester has been tested according to

DIN/VDE 871, Parts 2 and 20 (comparable with CISPR) and the telecommunications order 1046/84 of the German Postal Services.

If the protocol tester is operated together with other systems, particular attention must be paid to the use of sufficiently screened connection cables.

3.2 Manufacturer's Certificate

(see overleaf)

SIEMENS

Bescheinigung des Herstellers/Importeurs

Hiermit wird bescheinigt, daß der/die/das

Protokolltester K1197

(Gerät, Typ, Bezeichnung)

In Übereinstimmung mit den Bestimmungen der

Verfügung Fernmeldewesen 1046/84

(Amtsblattverfügung)

funk-entstört ist.

Der Deutschen Bundespost wurde das Inverkehrbringen dieses Gerätes angezeigt und die Berechtigung zur Überprüfung der Serie auf Einhaltung der Bestimmungen eingeräumt.

Siemens AG, Meßgerätewerk Berlin

Namen des Herstellers/Importeurs



ISDN BAI2 INTERFACE

Protocol Tester K1197

SIEMENS AG

Date: : February, 1990

1	Introduction	1-1
1.1	General	1-1
1.2	Features	1-1
1.3	Replacing the interface board	1-2
1.4	Position of the BAI2 boards in the K1197	1-3
1.5	Jumpers and their positions	1-4
1.6	Status of the LEDs	1-7
2	Basics	2-1
2.1	IOM interface	2-1
2.1.1	IOM1	2-1
2.1.2	IOM2	2-2
3	Configurations	3-1
3.1	Monitor	3-1
3.2	NT emulation	3-2
3.3	TE emulation	3-3
4	Interface board	4-1
4.1	Description	4-2
4.1.1	ASIC 29	4-2
4.1.2	ARCOFIs	4-3
4.1.3	PLL	4-3
4.1.4	DC/DC converter	4-4
4.2	Configuration of RAM and I/O address space	4-5
4.3	Connectors	4-6
4.3.1	Functional overview of connector X3	4-9
4.4	Electrical characteristics	4-11
5	ISDN-So connector module	5-1
5.1	Introduction	5-1
5.2	Installation	5-1
5.3	Electrical characteristics	5-1
5.4	Front view	5-2
5.5	Connectors	5-3

5.5.1	Line connectors	5-3
5.5.2	Auxiliary connectors	5-4
5.5.2.1	External clock	5-4
5.5.2.2	Power feeding	5-4
5.5.2.3	External B channel access	5-5
5.6	Principles of transmission	5-6
5.6.1	Configurations	5-6
5.6.1.1	Point-to-multipoint (passive bus)	5-6
5.6.1.2	Point-to-point	5-7
5.6.2	Framestructure at the S/T reference point	5-7
5.7	Hardware description	5-8
5.8	LEDs for line status	5-8
6	ISDN-Upo connector module	6-1
6.1	Introduction	6-1
6.2	Installation	6-1
6.3	Electrical characteristics	6-1
6.4	Front view	6-2
6.5	Connectors	6-3
6.5.1	Line connectors	6-3
6.5.2	Auxiliary connectors	6-3
6.5.2.1	Power feeding	6-3
6.5.2.2	External B channel access	6-4
6.6	Principles of transmission	6-5
6.7	Hardware description	6-5
7	Literature	7-1

1 INTRODUCTION

1.1 General

This documentation describes the basic functions of the BAI2 board (**B**asic-**A**ccess-**I**nterface) for the SIEMENS protocol tester K1197.

This documentation includes the summarized technical data also.

1.2 Features

The most significant features of the BAI2 board are listed below.

1. 16 MHz CPU (68000)
2. 2 MByte local RAM
3. ISDN interfaces S₀ and U_{p0} by changing connector modules only
4. Line clock accuracy in NT/LT emulation mode (clock master) ± 5 ppm
5. External input for line clock supply 15.36 MHz
6. Integrated DC/DC converter for phantom feeding
7. Integrated codecs (A/ μ -law) for 4 kHz voice services
8. Clock synchronization and data transfer between two interface boards
9. Programmable B channel routing
10. Generation of CRC errors in D and B channel

1.3 Replacing the interface board



CAUTION

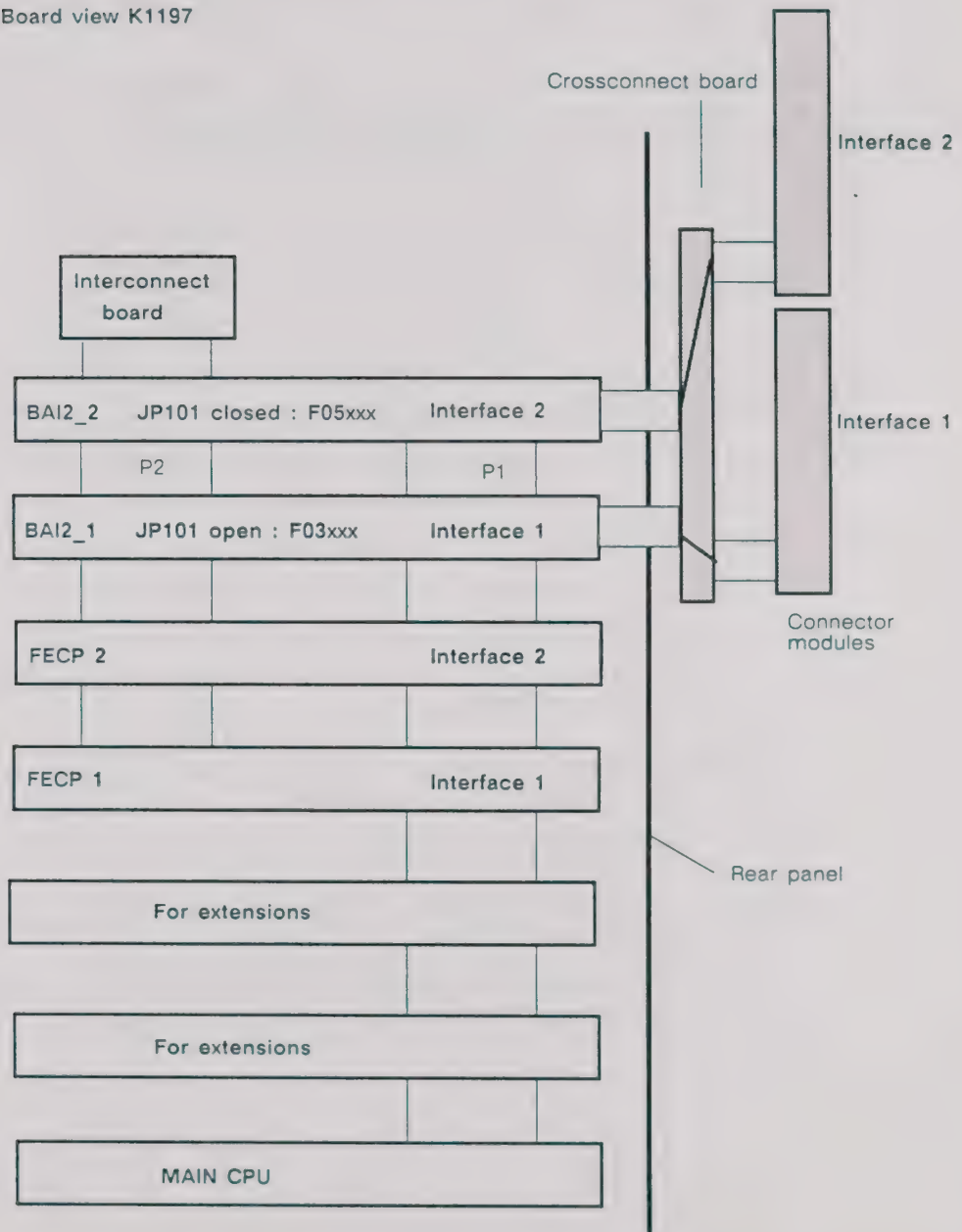
When handling the board, the regulations for electrostatic sensitive devices (ESD) must be observed.

Proceed as follows to replace the interface board :

1. Switch off K1197
2. **Disconnect mains plug from device**
3. Remove connector modules (4 screws) and crossconnect board (4 screws)
4. Remove housing cover (4 screws)
5. Pull out interface board by using the red handles
6. Insert new interface board(s)
7. Reinstall crossconnect board and connector module(s)
8. Screw back housing cover
9. Reconnect mains plug and switch device on

1.4 Position of the BAI2 boards in the K1197

Board view K1197



If two BAI2 boards are installed you should select the addresses according to the picture, if only one is needed install it in the slot of interface 1 and J101 should be open.

1.5 Jumpers and their positions

The BAI2 board contains several jumpers for configuration of the board. The table below shows an functional overview while the positions of the jumpers can be found on one of the following pages.

Jumper	Functional description	Default
JP101	Selects the board addresses of RAM and I/O	open
JP104-107	For production test only	all installed
JP109-111	For production test only	all installed
JP112-114	Selects bus priority 1 or 0 (Pin 1-2)	Conn. Pin 2-3
JP201	Inhibit of CPU start	installed
JP604,604	Switch between IOM1 and IOM2 (Pin 2-3)	Conn. Pin 1-2
JP702-704	Switch between IOM1 and IOM2 (Pin 2-3)	Conn. Pin 1-2
JP801	Selects CPU clock 16 or 8 MHz (Pin 2-3)	Conn. Pin 1-2

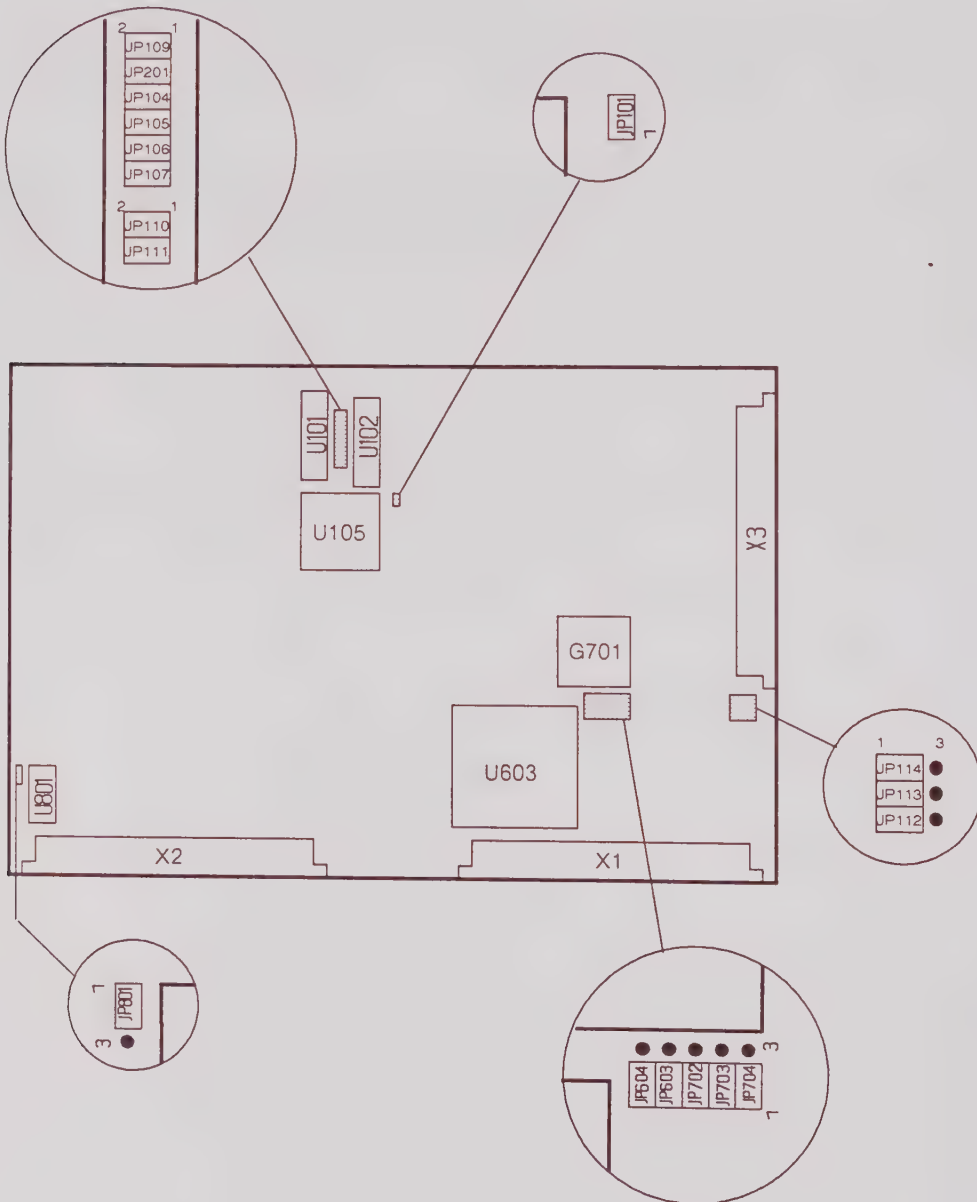
1. CPU clock speed: The CPU can run with 8 or 16 MHz , this is selected by JP801.

- o For 8 MHz operation the pins 2-3 have to be connected.
- o For 16 MHz operation the pins 1-2 have to be connected.
- o **Note:** The CPU does not work if JP801 is not installed. If 16 MHz is selected, check if CPU is a 16 MHz type.

2. Address space: Two basic addresses can be selected by JP101 for the parallel operation of two BAI2 boards in one K1197.

- o If the jumper is installed the I/O addresses start at F05000 and RAM addresses start at 400000 (- 5FFFFFF).
- o If the jumper is not installed the I/O addresses start at F03000 and RAM addresses start at 600000 (- 7FFFFFF).
- o **Note:** If the jumper is lost there are some extra ones in a small plastic bag inside the housing of K1197.

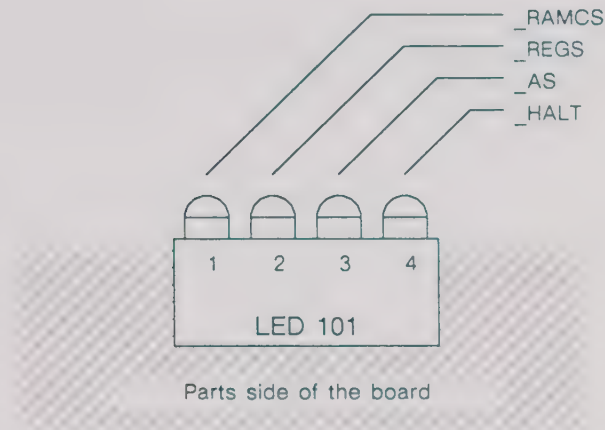
3. **Start register:** The operation of the processor can be stopped by removing the jumper JP201.
 - o The start register can only be activated if this jumper is installed.
4. **Board test:** The jumpers JP104 to JP107 and JP109 to JP111 are for test purposes only. These jumpers should be installed in normal operation. If the jumpers JP104 to JP107 are open the VME bus drivers are inactive. If the jumpers JP109 to JP111 are open the CPU is in high-impedance state.
5. **Bus priority:** The bus priority can be selected by the jumpers JP112 to JP114.
 - o For the priority 0 **all three** jumpers should connect Pin 1-2.
 - o For the priority 1 **all three** jumpers should connect Pin 2-3.
6. **IOM1/IOM2 switch:** The ISDN chips on the BAI2 board can be configured in IOM1 or IOM2 mode by the jumpers JP603, JP604 und JP702 to JP704.
 - o For IOM1 mode **all five** Jumpers should connect Pin 1-2.
 - o For IOM2 mode **all five** Jumpers should connect Pin 2-3.



Positions of jumpers on BAI2 board

1.6 Status of the LEDs

At the top of the board four LEDs are located which display control signals. The LEDs are illuminated if their signal is active ("low").



LED Display

With the help of the LEDs you can make fast decisions whether your board is operating correct.

1. _RAMCS: This LED is active if the local CPU or another VME bus master is accessing the onboard RAM.
2. _REGS: This LED is active if one of the peripherals of the BAI2 board is accessed.
3. _AS: This LED is active if the local CPU is active or the board is accessed by another VME bus master.
4. _HALT: This LED is active if the local CPU is halted. This could happen after a double bus error or after reset before the board is booted. Once the board is booted and started this LED should not be active.

2 BASICS

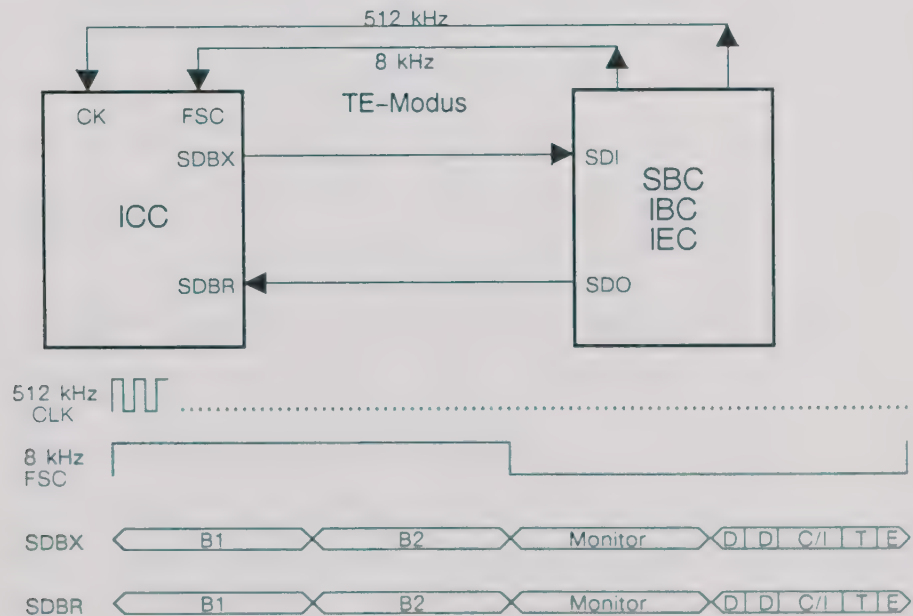
2.1 IOM interface

The **IOM** interface (**ISDN Oriented Modular**) is an interface to a family of **SIEMENS ISDN ICs** (**SBC**, **IEC**, **IBC**). It exists in two versions, **IOM Rev.1** (called **IOM1**) and **IOM Rev.2** (called **IOM2** or **GPI=General Purpose Interface**).

The **IOM** interface handles the data transfer from the **ICC** (**ISDN Communication Controller**) on the **BAI2** board to the layer 1 ICs (**SBC**, **IEC**, **IBC**) on the connector module. Data are transferred in **NRZ** code (**Non Return to Zero**).

2.1.1 IOM1

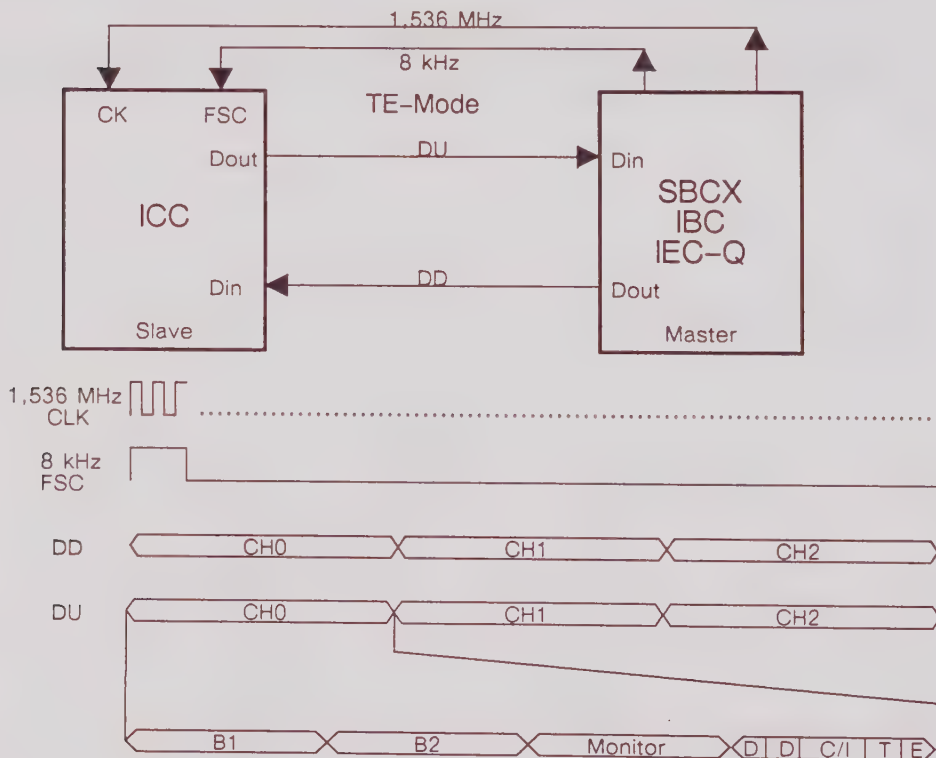
This interface has a bit clock of 512 kBit/s. The following functional description gives you an overview of this configuration.



Functional description of IOM1 interface

2.1.2 IOM2

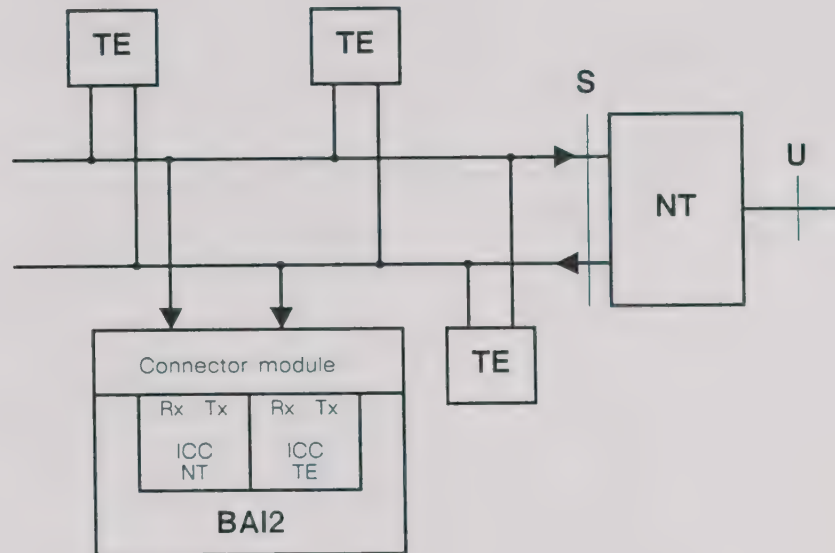
The IOM2 interface has a higher transfer rate than the IOM1 interface. IOM2 is an international standard supported by SIEMENS, AMD, VALVO and more. The IOM2 interface can be used with bit clocks of 512 kHz (one channel = same as IOM1), 1.536 MHz (multiplexed three channels, TE mode) or 8.192 MHz (multiplexed 16 channels, NT mode).



Functional description of IOM2 interface in the TE mode

3 CONFIGURATIONS

3.1 Monitor

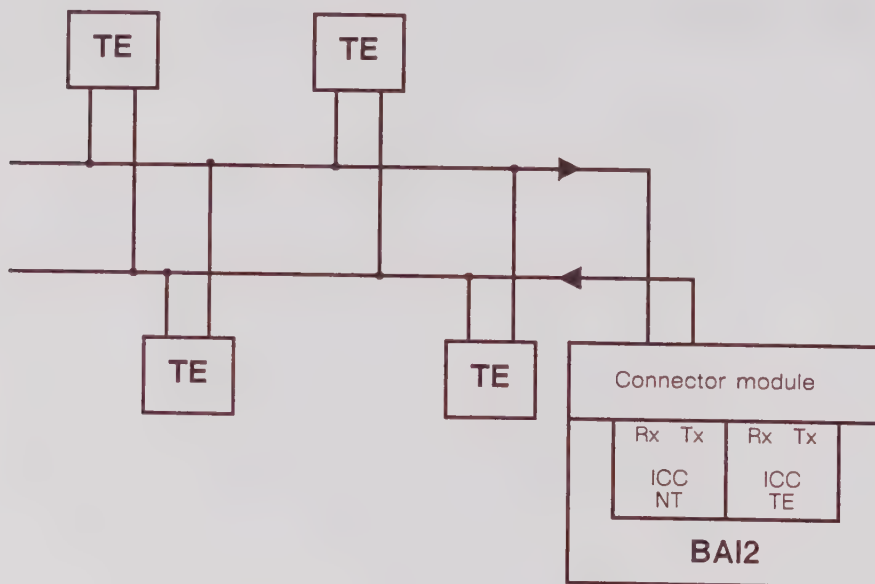


BAI2 board with S_0 connector module in monitor mode

In monitor mode two receivers (one configured as NT, one as TE) receive both directions. This configuration has the impedance of one TE but does not influence the line in any way.

The receive clocks are generated by a PLL circuit in the TE receiver.

3.2 NT emulation

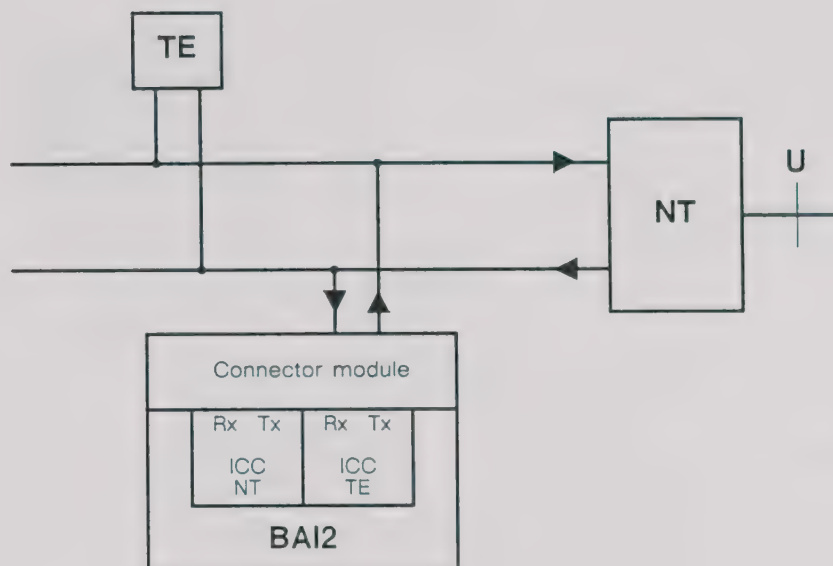


BAI2 board with S_0 connector module in NT emulation mode

In NT emulation mode the BAI2 board is emulating a Network Termination (NT or LT). For voice services two codecs are implemented on the BAI2 board which can be used in any emulation mode. If you want to feed TEs select internal or external power feeding. You can also select the origin of your line clock, internal, external or from the second interface board.

The line clocks can be generated by the internal 15.36 MHz oscillator, from a second interface board or from an external (via connector module) high accuracy 15.36 MHz clock.

3.3 TE emulation

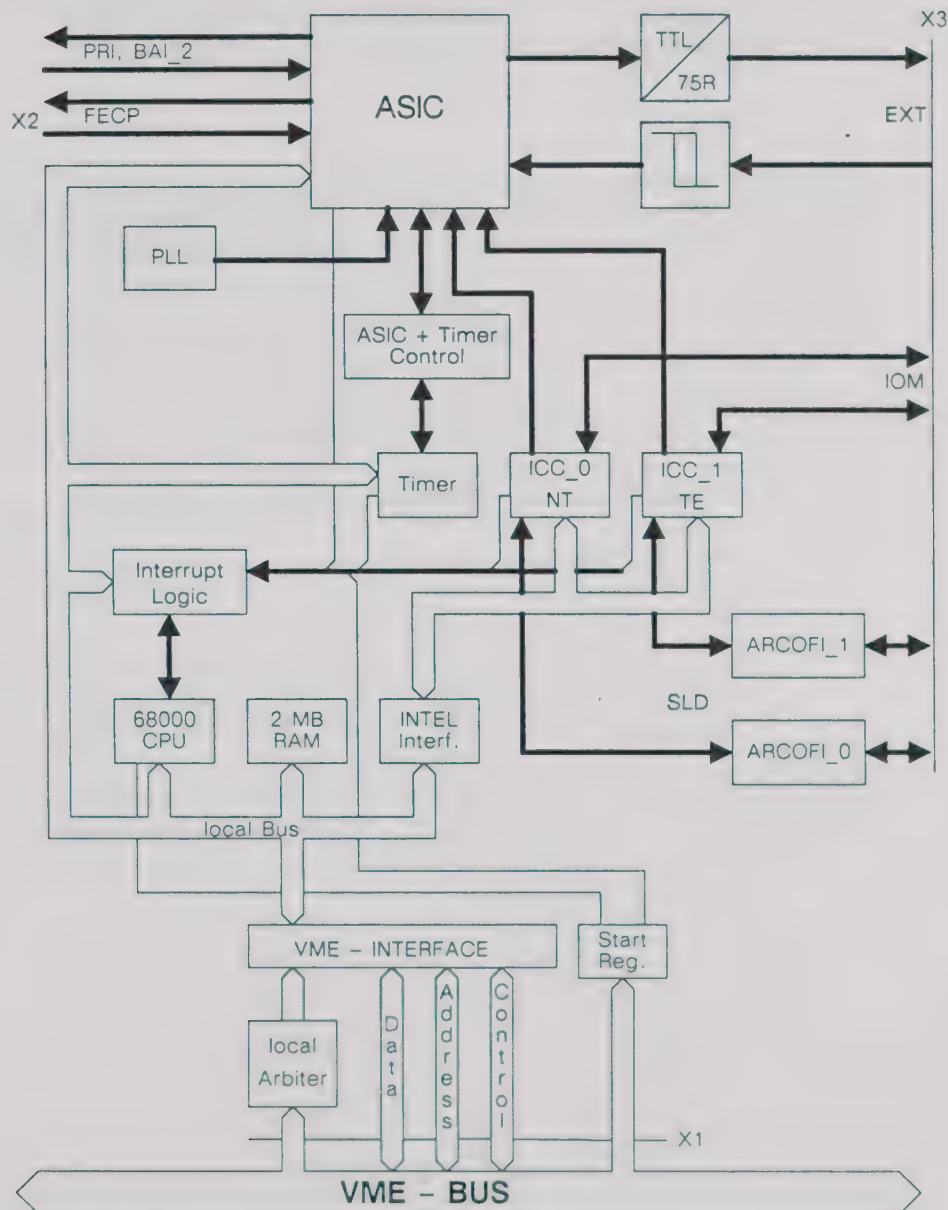


BAI2 board with S_0 connector module in TE emulation mode

In TE emulation mode the BAI2 board is emulating a Terminal Equipment (TE). For voice services two codecs are implemented on the BAI2 board which can be used in any emulation mode.

The receive clocks are synchronized to the line.

4 INTERFACE BOARD



Design of the BAI2 board

4.1 Description

The BAI2 board is a multilayer (six layers) board in VME bus format. The BAI2 board contains its own CPU (68000) with 2 MByte of local RAM.

The connector X1 connects the board to the VME bus, X2 connects it to the frontend processor (FECP) and X3 (on the small side) connects it to the connector module.

Some parts of the board and their features are described here below.

4.1.1 ASIC 29

This ASIC was developed especially for the complex interworking of data and clock routing from the ICCs to the FECP and the external or internal sources. It contains some interface logic to make different data formats familiar with each other. Together with a discrete PLL circuit it enables data transfer from one interface board to the other in a special burst mode.

It is possible to modify the CRC bytes of B or D channel transmitters with an internal logic of the ASIC. This feature helps you to test the error handling of a NT or TE.

The ASIC contains also a matrix for B channel data with 11 inputs and 13 outputs. Each output can be switched to every input. There are four groups of in- and outputs.

- o Interface to the FECP for B channel data with a programmable inversion of data.
- o Interface to the serial port A of the ICC with a conversion from 128 kBit/s to 64 kBit/s and vice versa.
- o Interface to the external B channel access connector on the connector module.
- o Interface for data exchange with a second interface board (PRI or BAI2) using the burst mode.

For the configuration of the BAI2 board and the connector modules the ASIC contains a PIO (Parallel Input/Output). The PIO has two 8 Bit Output Ports (Port A and Port B) and one 4 Bit Input Port (Port C).

4.1.2 ARCOFIs

With the codec function of the two onboard ARCOFIs (Audio Ringing Codec Filter) it is possible to make a telephone call in emulation mode or listening to a call in monitor mode. The de/encoding is programmable for A and μ -law. For using this features you need a handset with a male 15-pin Cannon connector (for pinout see chapter 5) which can be connected to the connector module. Tones for busy, ringing and for test tones (sine) can be generated by the codecs.

4.1.3 PLL

Onboard an analogue PLL (Phase Locked Loop) is located. With this PLL the BAI2 board can be synchronized to frames from the second interface board via the frame sync signals. In this mode the board acts as a clock slave the second interface board acts as clock master.

NOTE: After programming the slave mode by the menu the PLL needs up to 20 seconds until PLL is locked and clocks are synchronized.
A BAI2 board is unable to work if it is in slave mode and the second interface board is not programmed to be clock master.
If external clock is programmed and no correct external clock (15.36 MHz) is present at the BNC input on the connector module the BAI2 board will fail to work.
If two interface boards are synchronized and no data can be passed check the configuration of the **DIP switches** on the **crossconnect board**.

4.1.4 DC/DC converter

The BAI2 board contains a DC/DC converter for the power feeding of terminals (power source 1, phantom) with a voltage of 40 VDC and with up to 120 mA current. This voltage is decoupled from all other voltages on the interface board.

The DC output is protected against short circuit and reverse voltages by a reverse diode and a PTC resistor. If the current is increased to more than 120 mA it will be limited by the the PTC resistor. If a reverse voltage is present on the line the diode limits this voltage at the output of the converter and the PTC will limit the current to a value unable to destroy circuitry.

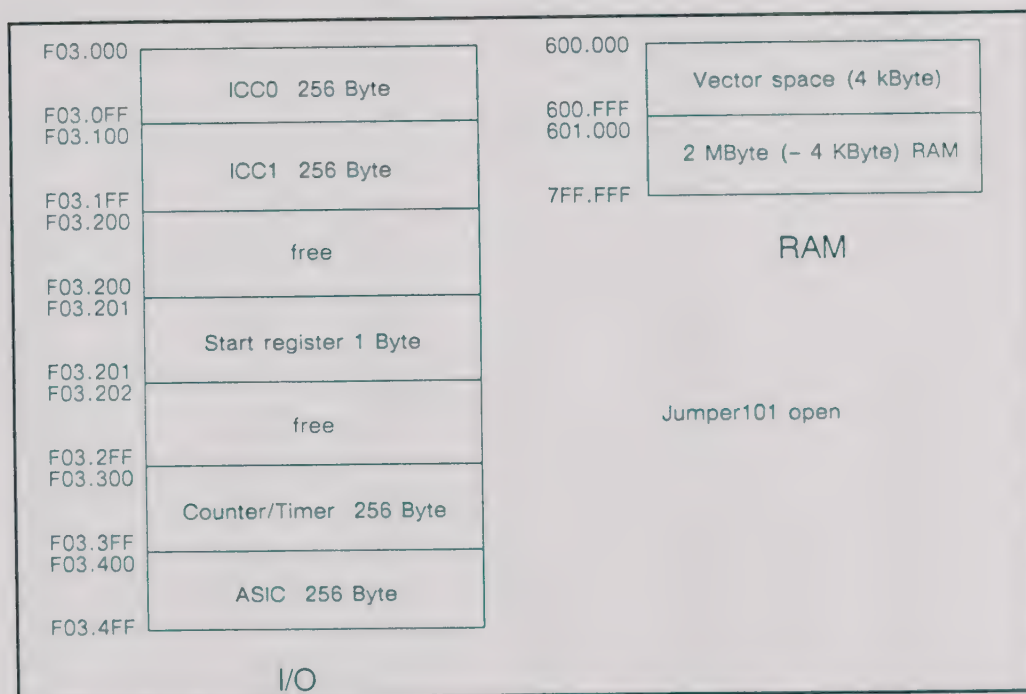
After a short circuit was recognized it takes some time for the PTC resistor to recover its impedance.

The output voltage of the DC output can be programmed to be reversed thus enabling a emergency supply (power feeding from the net in case of a shutdown of local power supply) of a telephone.

4.2 Configuration of RAM and I/O address space

Function	Jumper JP101 not installed		Jumper JP101 installed	
	Start address	End address	Start address	End address
RAM	600.000	7FF.FFF	400.000	5FF.FFF
I/O	F03.000	F03.4FF	F05.000	F05.4FF

Address space of BAI2 boards



Configuration of address space

Due to an optimal address decoding some I/O devices have more reserved space than they actually need.

4.3 Connectors

Pin Number	Row A Signal Descriptor	Row B Signal Descriptor	Row C Signal Descriptor
1	DV0	_BBSY	DV8
2	DV1	_BCLR	DV9
3	DV2	NC	DV10
4	DV3	_BG0IN	DV11
5	DV4	_BG0OUT	DV12
6	DV5	_BG1IN	DV13
7	DV6	_BG1OUT	DV14
8	DV7	NC	DV15
9	GND	NC	GND
10	YSCLK	NC	NC
11	GND	NC	_BERRV
12	_DS1	_BR0	_SYSRESET
13	_DS0	_BR1	_LWORD
14	_WRITEV	NC	AM5
15	GND	NC	AV23
16	_DTACKV	AM0	AV22
17	GND	AM1	AV21
18	_ASV	AM2	AV20
19	GND	AM3	AV19
20	_IACKV	GND	AV18
21	NC	NC	AV17
22	NC	NC	AV16
23	AM4	GNDA	AV15
24	AV7	NC	AV14
25	AV6	NC	AV13
26	AV5	NC	AV12
27	AV4	NC	AV11
28	AV3	NC	AV10
29	AV2	NC	AV9
30	AV1	NC	AV8
31	-12V	NC	+12V
32	+5V	+5V	+5V

Connector X1

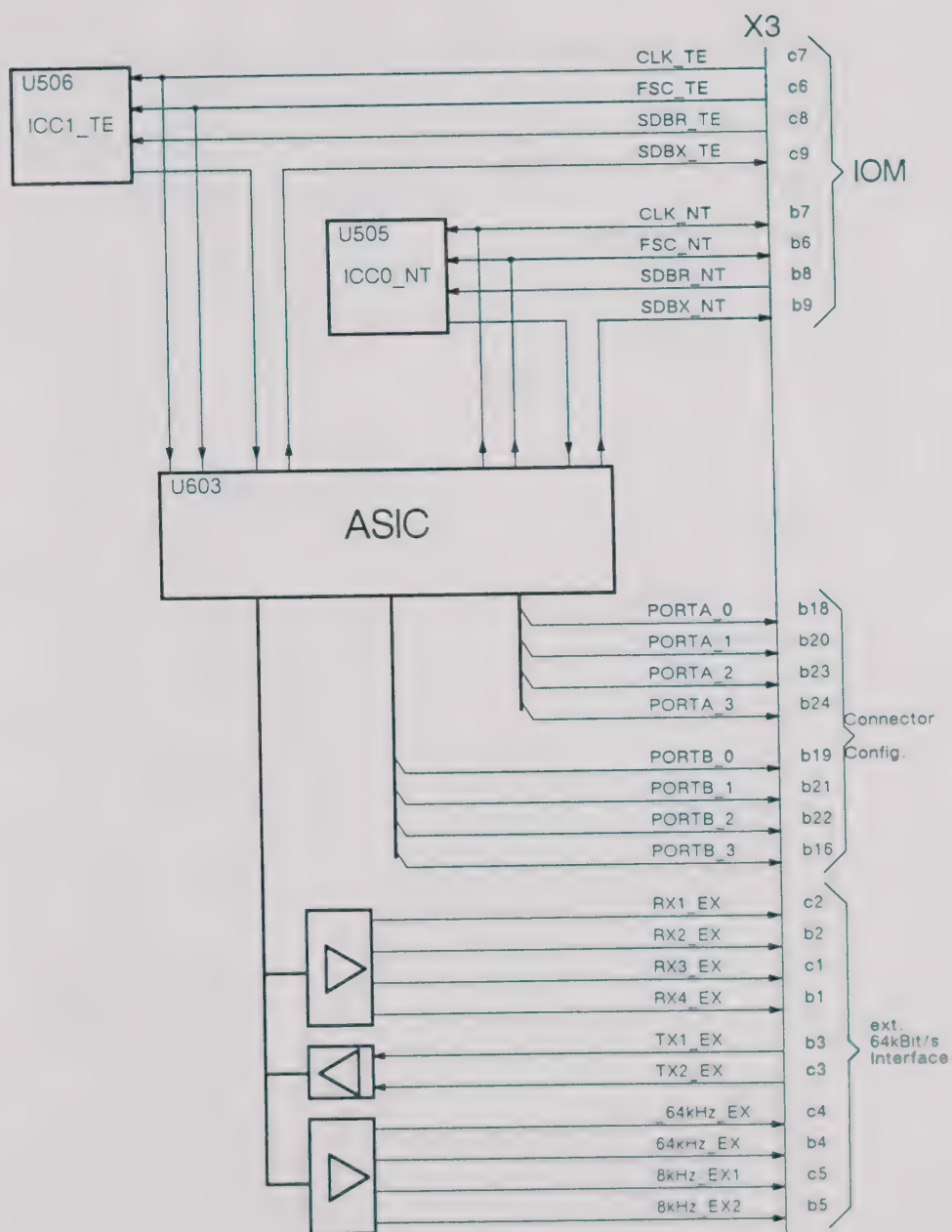
Pin Number	Row A Signal Descriptor	Row B Signal Descriptor	Row C Signal Descriptor
1	NC	+5V	NC
2	NC	GND	NC
3	NC	NC	NC
4	NC	NC	NC
5	NC	NC	NC
6	SRX1	NC	NC
7	TxC	NC	NC
8	RxC	NC	SRX3
9	NC	NC	RxC
10	NC	NC	NC
11	NC	NC	RxC
12	NC	GND	SRX4
13	NC	+5V	RxC
14	NC	NC	NC
15	NC	NC	NC
16	NC	NC	NC
17	STX2	NC	NC
18	NC	NC	NC
19	NC	NC	NC
20	NC	NC	NC
21	NC	NC	NC
22	NC	NC	NC
23	NC	NC	NC
24	NC	NC	NC
25	NC	NC	NC
26	NC	NC	NC
27	NC	NC	NC
28	STX1	NC	NC
29	TxC	NC	NC
30	NC	NC	NC
31	NC	GND	NC
32	NC	5V	SRX2

Connector X2

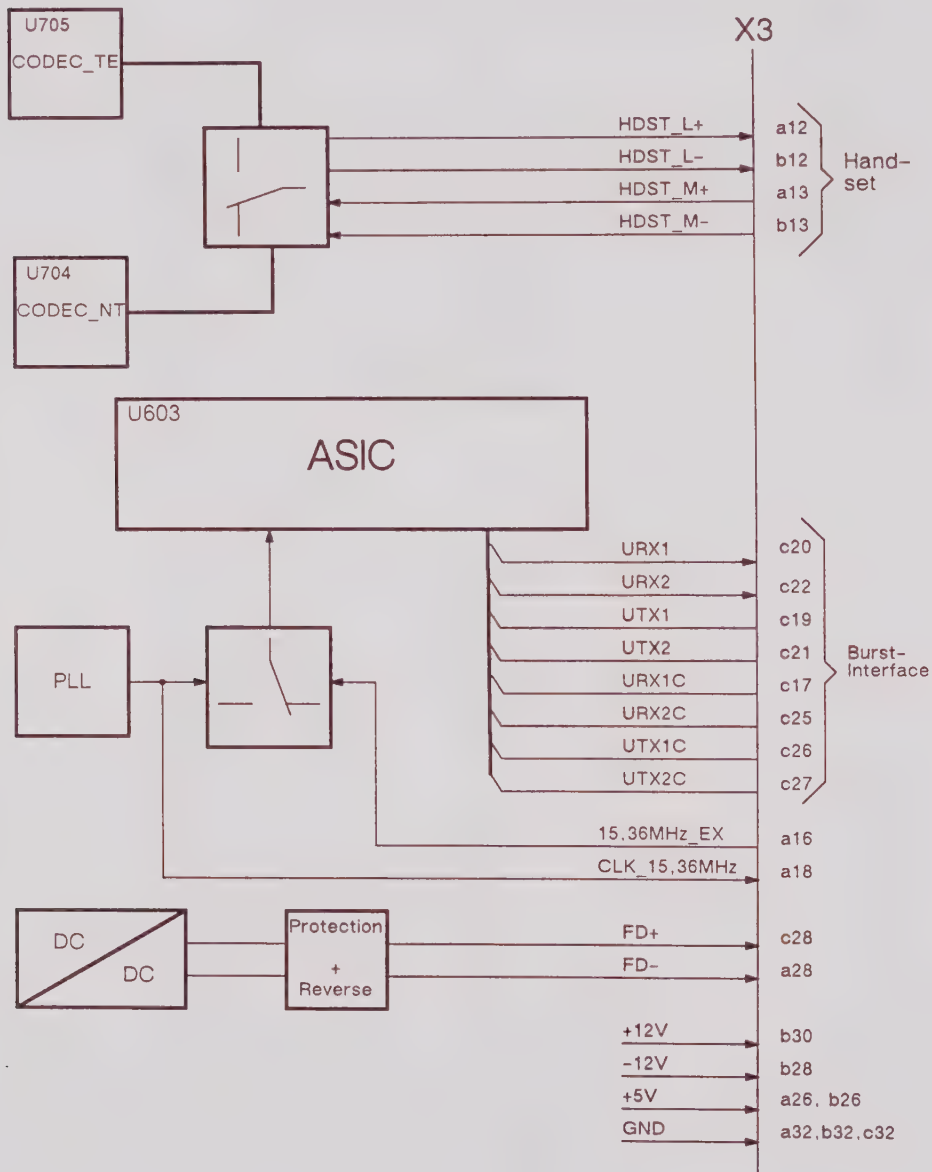
Pin Number	Row A Signal Descriptor	Row B Signal Descriptor	Row C Signal Descriptor
1	X3_a1	RX4_EX	RX3_EX
2	X3_a2	RX2_EX	RX1_EX
3	X3_a3	TX1_EX	TX2_EX
4	X3_a4	64kHz_EX	_64kHz_EX
5	NC	8kHz_EX2	8kHz_EX1
6	NC	FSC_NTB	FSC_TE
7	NC	CLK_NTB	CLK_TE
8	NC	SDBR_NT	SDBR_TE
9	NC	SDBX_NT	SDBX_TE
10	NC	NC	NC
11	NC	NC	NC
12	HDST_L+	HDST_L-	AGND
13	HDST_M+	HDST_M-	NC
14	NC	NC	NC
15	NC	NC	NC
16	15,36MHz_EX	X3_b16	NC
17	NC	_64kHz_EX	URX1C
18	CLK_15,36MHz	X3_b18	NC
19	SYNC_EX	X3_b19	UTX1
20	CLK_16MHz	X3_b20	URX1B
21	FSC_NTB	X3_b21	UTX2
22	NC	X3_b22	URX2B
23	NC	X3_b23	64kHz_EX
24	NC	X3_b24	_RESET
25	X3_a25	_64kHz_EX	URX2C
26	+5V	+5V	UTX1C
27	NC	NC	UTX2C
28	FD-	-12V	FD+
29	NC	NC	NC
30	CLK_NTOC	+12V	_CONNINST
31	NC	NC	64kHz_EX
32	GND	GND	GND

Connector X3

4.3.1 Functional overview of connector X3



Signals to the connector module (part 1)



Signals to the connector module (part 2)

4.4 Electrical characteristics

Supply ratings : + 5 VDC 2,1 A typ.
 +12 VDC 0,05 A typ. (without phantom feeding)
 -12 VDC 0,05 A typ.

DC/DC converter : 5 W, 40 VDC +5/-15%, 120 mA

Accuracy of clock oscillator : 15,36 MHz $\pm$ 5 ppm

Operating temperature : 0 - 70 °C (32 - 160 Degrees Fahrenheit)

5 ISDN-S₀ CONNECTOR MODULE

5.1 Introduction

The ISDN-S₀ connector module is designed as a plug-in unit of the K1197 and is used with the BAI2 board only. The connector module is physically connected to the interface board (via the crossconnect board) by a 96-pin DIN connector, including two separate IOM interfaces plus signals for control and supply.

The connector module gives access to four-wire ISDN lines at reference point S/T.

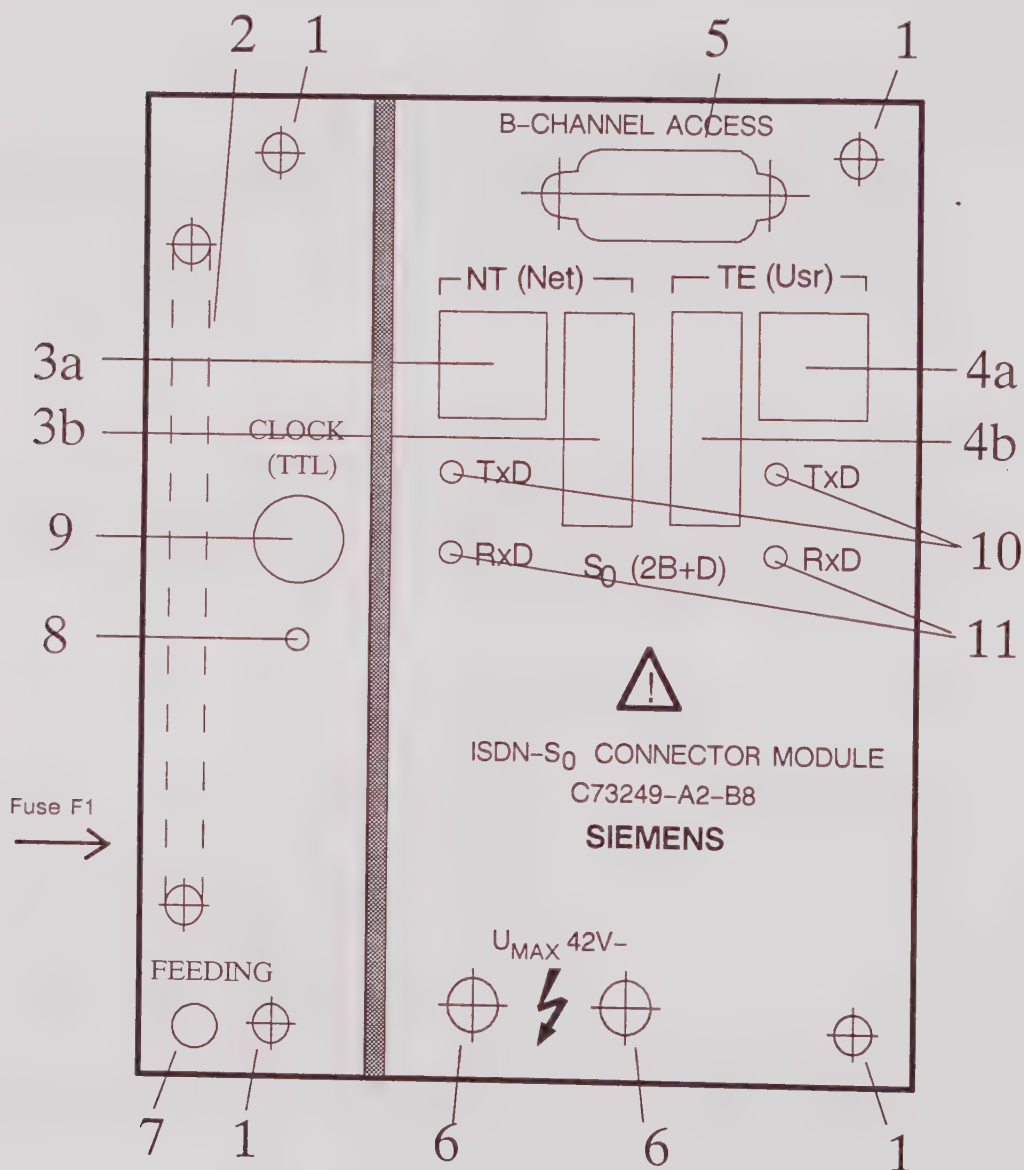
5.2 Installation

Switch off the power of the K1197. Make sure that the interface boards and the crossconnect board are installed correctly. Push the connector module into the corresponding slot of the BAI2 board on the back side of the K1197. Be careful not to damage one of the pins of the male 96-pin connector on the crossconnect or interface board. Fasten the connector module with four screws (pos. 1). For removal use the handle (pos. 2).

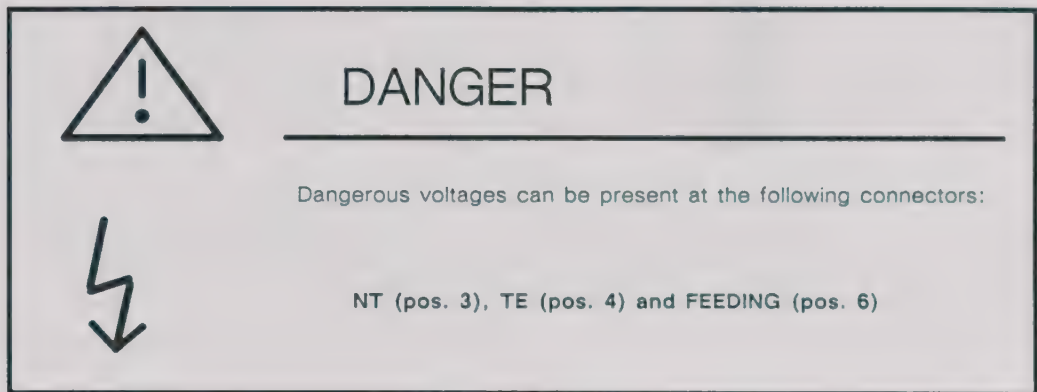
5.3 Electrical characteristics

Supply ratings :	5 VDC, less than 250 mA
External connectors :	2 x RJ-45, 2 x TAE 8+4
	according to CCITT I.430 when using the included cord
	(length 3 meters)
	protected against spikes up to 1 kV
	1 x 2 pin (for 4 mm plugs) white (+) and black (-)
	for power feeding sink/source,
	maximum ratings: 42 VDC, 120 mA
	1 x female 15-pin Cannon for B channel access,
	1 x BNC for external clock 15,36 MHz
Accuracy of NT clocks :	+/- 5 ppm
Operating temperature :	0 - 50 degrees Celsius (32 -125 degrees Fahrenheit)

5.4 Front view



5.5 Connectors



For prevention of electrical shock the red LED (pos. 7) senses the line and indicates when a voltage greater than 20 VDC is present.

5.5.1 Line connectors

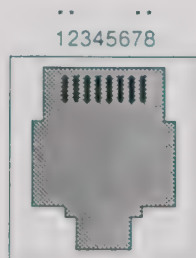
There are two connectors dedicated as *from* Network NT (Net) (pos. 3a and b) and *from* Terminal Equipment TE (Usr) (pos. 4a and b).

There are three operating modes of the K1197:

TE emulation: Connect your NT to the NT (Net) connector. The connector module can be programmed by the menu software to a bus termination by 100 ohms or high impedance (S-Bus).

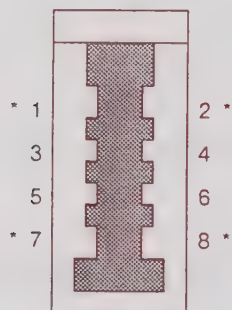
NT emulation: Connect your TE to the TE (Usr) connector. The connector module terminates the line with 100 ohms.

Monitor: Deactivate the line. Connect your line to one of the four connectors (pos. 3 or 4) which are connected parallel in this operating mode. Activate the line from NT or TE. The connector module does not influence the line in this mode.



Female connector
front view

Pin Mode	Power Polarity	T1D1	
		NT	TE
Tx +	-	4	3
Tx -	-	5	6
Rx +	+	3	4
Rx -	+	6	5



Female connector, front view

Pin Mode	Power Polarity	FTZ 1R230 FTZ 386TR2	
		NT	TE
Tx +	-	3	6
Tx -	-	4	5
Rx +	+	6	3
Rx -	+	5	4

Note: Pins marked with a asterisk are not connected !

5.5.2 Auxiliary connectors

5.5.2.1 External clock

An external 15.36 MHz clock can be fed into the BNC-connector (pos. 9). This is a TTL-level input terminated by a 75 ohms resistor. The activation of this clock source by the software menu is indicated by the yellow LED (pos. 8), if no clock is present the board is not able to work.

5.5.2.2 Power feeding

In the mode NT emulation the internal phantom power feeding can be activated. For feeding by an external source and for access to the feeding voltage there are two connectors for 4 mm plugs white (+) and black (-) (pos. 6). According to the specification for S₀ lines the recommended voltage for feeding an TE is 40 VDC (+2/ -6 VDC). The red LED (pos. 7) indicates a feeding on the line.

The use of these connectors is different in the three modes.

TE emulation: If a voltage for feeding the TEs is generated by the NT, the voltage can be shunted and/or accessed by a voltmeter.

NT emulation: If the internal power source is selected in configuration menu its voltage can be accessed (if no voltage is present check two jumpers X9, X10). If external power source is selected you can feed your TEs by connecting an external source. To prevent a possible damage of the the transformers the maximum current is limited to 160 mA by an internal fuse (F1) in the connector module. If the fuse is blown it can be easily replaced (see view of connector module for location).

Monitor: same as TE emulation

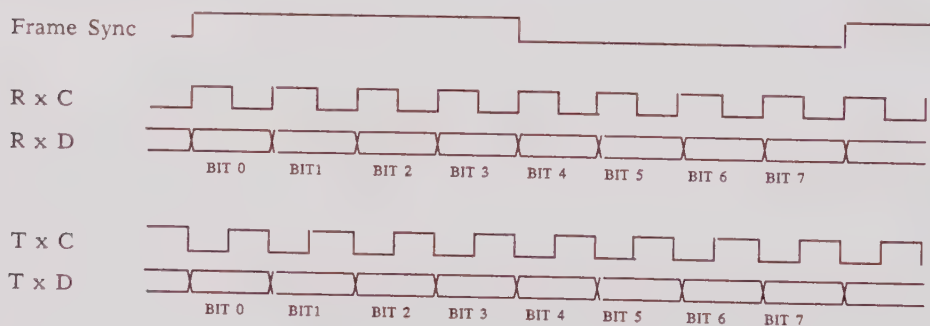
5.5.2.3 External B channel access

The B channels can be accessed by a female 15-pin connector (pos. 5).
There are two types of data:

- 1.) 64 kBit/s data streams with clocks, TTL level respective to pin 15
 - 2.) Voice audio signals for a handset:
- Symmetric, sensitivity (programmable) -20 dB, bandwidth 200 - 3500 Hz

Pin 1:	Output R x D / B1 from Net
Pin 2:	Output (+) voice audio
Pin 3:	Output R x D / B2 from U _{sr}
Pin 4:	Input T x D / B2 (use only in emulation mode)
Pin 5:	Input (+) voice audio
Pin 6:	Output R x C (64 kHz)
Pin 7:	Output Frame Sync B1 (8 kHz)
Pin 8:	Output Frame Sync B2 (8 kHz)
Pin 9:	Output R x D / B2 from Net
Pin 10:	Output R x D / B1 from U _{sr}
Pin 11:	Output (-) voice audio
Pin 12:	Input T x D / B1 (use only in emulation mode)
Pin 13:	Output T x C (64 kHz, inverted to R x C)
Pin 14:	Input (-) voice audio
Pin 15:	Digital ground

Timing diagrams

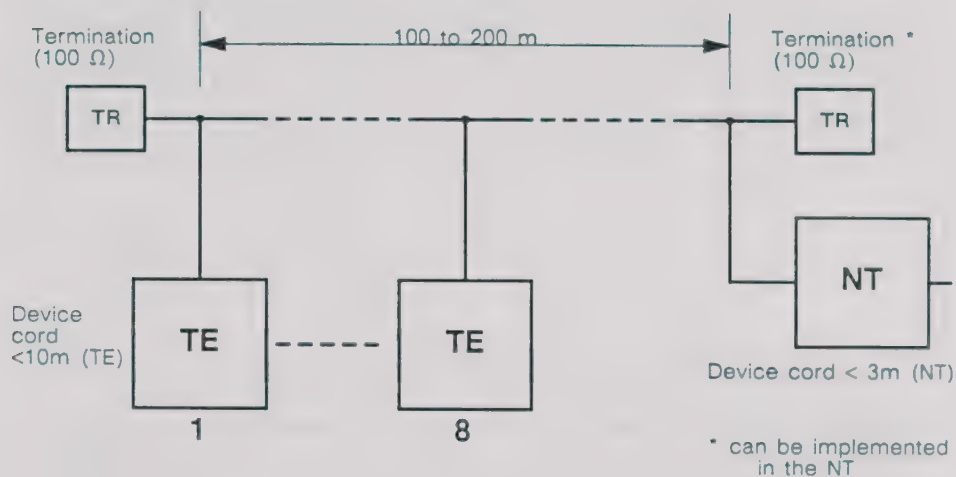


5.6 Principles of transmission

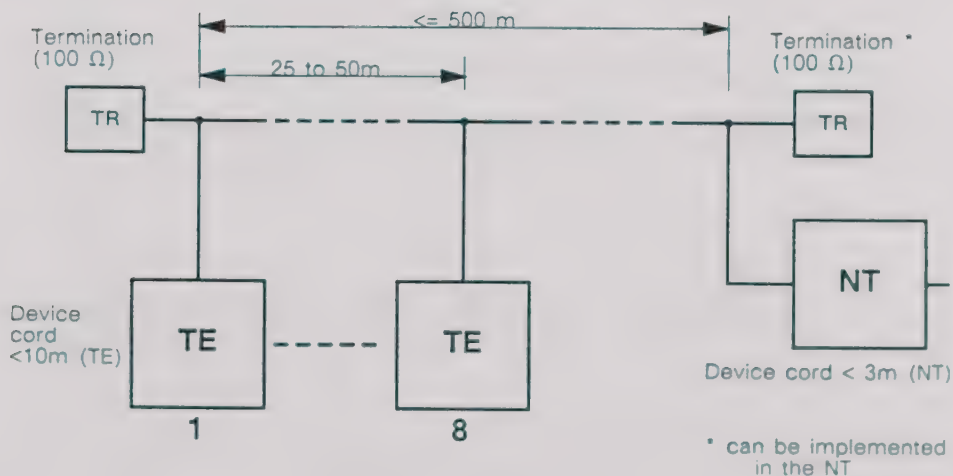
5.6.1 Configurations

5.6.1.1 Point-to-multipoint (passive bus)

The passive bus can be divided in two cases:

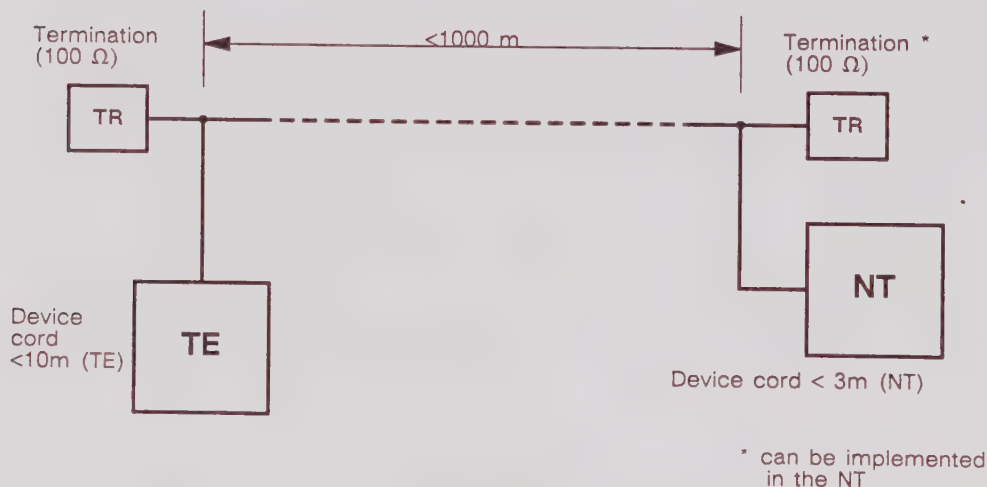


Short passive bus



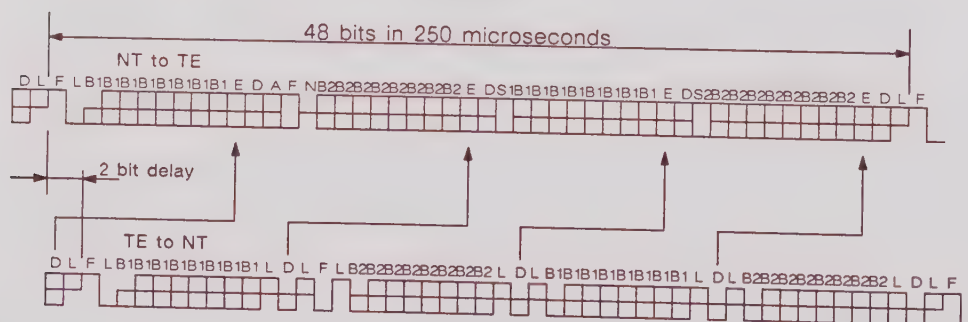
Extended passive bus

5.6.1.2 Point-to-point



5.6.2 Framestructure at the S/T reference point

The ISDN-S₀ connector module includes two SBC (or SBCX) chips providing full duplex four-wire data transmission of the 144 kBit/s user bit rate (2B + D) via normal telephone cables. Additionally bits for frame recovery, access control and DC free transmission are used totalling in a line bit rate of 192 kbit/s. The following picture shows the frame structure and the phase relations for both directions, the frame recovery is performed by a violation of AML code with two consecutive pulses of the same polarity.



The single bits have the following meaning:

- | | |
|----------|-----------------------------------|
| A: | Bit for the activation procedure |
| D,B1,B2: | Bits of the D,B1,B2 user channels |
| E: | D-channel echo bit |
| F: | Frame recovery bit |
| L: | Parity bit |
| N: | Control of the application |
| Sx: | Free bits |

5.7 Hardware description

The ISDN-S₀ connector module has two identical line interfaces. One is dedicated for the connection as a NT (X10,X11), the other for the connection as a TE (X12,X13). Both include overvoltage protectors (VD1 to VD4, V1 to V24), symmetry coils (L1,L2) and the line transformers (T1 to T4). Two SBC-Chips (D1,D2 PEB 2080 or D3,D4 PEB 2081, SIEMENS) provide a conversion from the line interface to the IOM interface (X3). The clocks for the SBCs and the supply voltages are generated by the BAI2 board (via X3). The comparator (D5) controls the lines and signals its status by four LEDs. The GAL 16V8 (D6) controls the switching of the clock source and the relays. The feeding voltage is limited by two VDR resistors (R6,R7) and rectified by four diodes (V28 to V31). The optocoupler (D4) decouples the signal "FEEDING" from the line and the red LED indicates when a feeding is present. The connectors for external feeding (X4) are current protected by a fuse (F1). The signals for the B channel access are only routed from the BAI2 board to the connector (X14). The jumper positions (J1,J2) are used for the internal phantom power feeding and need to be jumpered.

5.8 LEDs for line status

The physical status of the line is indicated by four green LEDs (pos. 10,11). Each LED has its own comparator on the respective pair of the line (RxD from TE, TxD to TE, RxD from NT, TxD to NT) checking the presence of an AC signal. This feature allows to check whether all wires are connected correctly, for example in state 'activated' both LEDs of the selected direction (*from* NT or *from* TE) are lighted.

6 ISDN-Upo CONNECTOR MODULE

6.1 Introduction

The ISDN- U_{po} connector module is designed as a plug-in unit of the K1195/K1197 and exists in two versions:

- 1.) C73249-A2-B6 is used with the ISDN-BRI (Type 2) in K1195 or with the BAI2 board in K1197. This version does not support voice services on B channel access connector.
- 2.) C73249-A2-B6-2 is used with the BAI2 board in K1197 only and supports voice and data on B channel access connector.

The connector module is physically connected to the interface board by a 96-pin DIN connector, including two separate IOM interfaces plus signals for control and supply.

The connector module gives access to two-wire ISDN lines at reference point U using the U_{po} (Ping-Pong) transmission mode.

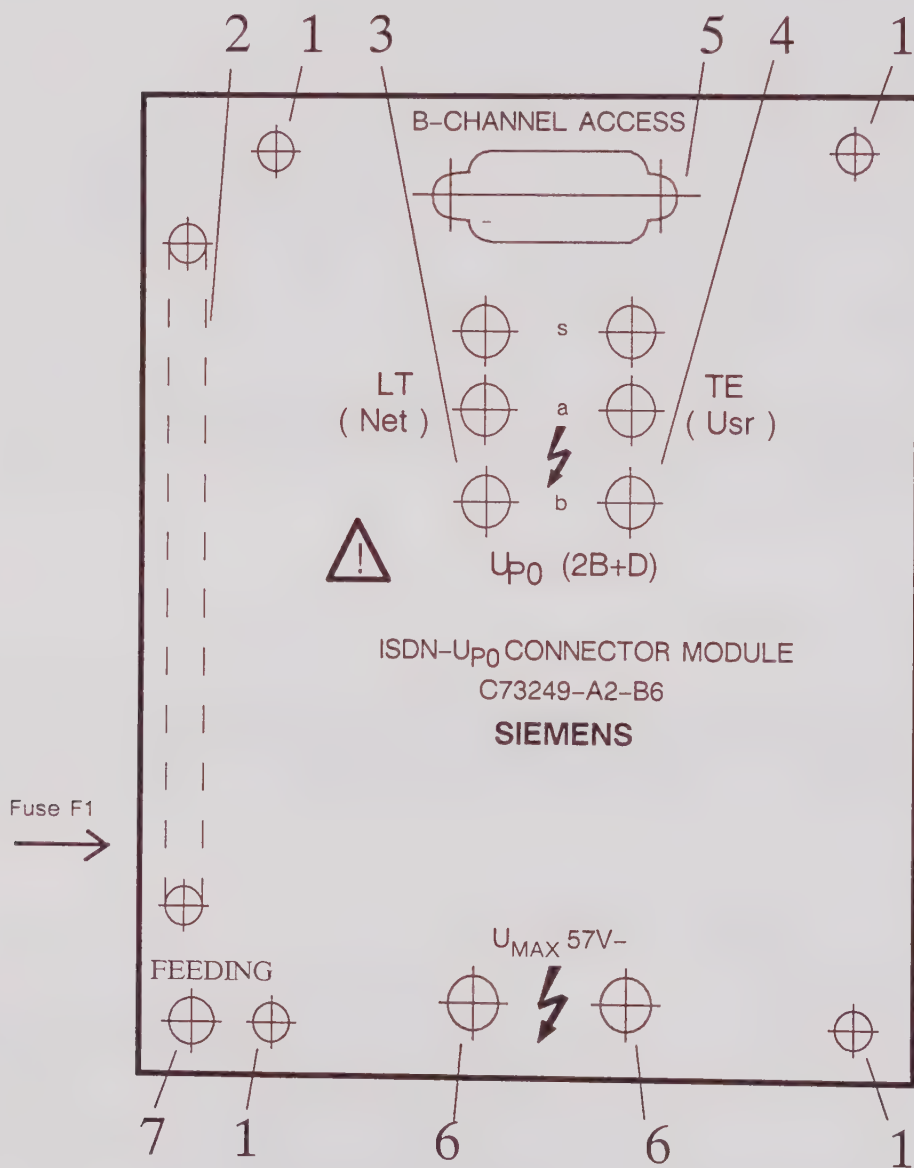
6.2 Installation

Switch off the power of the K1195/K1197. Make sure that the interface board (and crossconnect board) is installed correctly. Push the connector module into the slot on the back side of the K1195/K1197. Be careful not to damage one of the pins of the male 96-pin connector inside the device. Fasten the connector module with four screws (pos. 1). For removal use the handle (pos. 2).

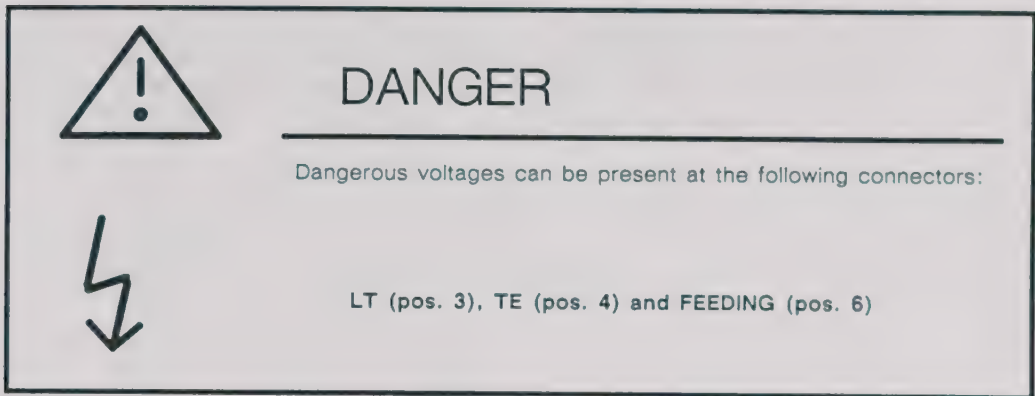
6.3 Electrical characteristics

Supply ratings :	5 VDC, less than 200 mA
External connectors :	2 x 3 pin (for 4 mm, banana plugs) for two-wire access according to ZVEI specification DKZ-N Part 1.2 May 1988 (ZVEI = Zentralverband der Elektroindustrie, West-Germany) protected against spikes up to 1 kV
	1 x 2 pin (for 4 mm, banana plugs) white (+) and black (-) for power feeding sink/source, maximum ratings: 57 VDC, 50 mA
	1 x female 15-pin Cannon for B channel access.
Accuracy of LT clocks :	± 50 ppm (version 1), ± 5 ppm (version 2)
Operating temperature :	0 - 50 °C (32 -125 degrees Fahrenheit)

6.4 Front view



6.5 Connectors



For prevention of electrical shock the red LED (pos. 7) senses the line and indicates when a voltage greater than 20 VDC is present.

6.5.1 Line connectors

There are two 3-pin connectors dedicated as *from* Line Terminator LT (Net) (pos. 3) and *from* Terminal Equipment TE (Usr) (pos. 4). Each pin is marked by a "s", "a" or "b". The "s" stands for shield, "a" and "b" are connected to the two-wire line. Both connectors are terminated by a 100 ohms resistor between "a" and "b".

There are three operating modes of the K1197:

TE Emulation: Connect your LT to the LT (Net) connector.

LT Emulation: Connect your TE to the TE (Usr) connector.

Monitor: Deactivate the line. Connect your LT to the LT (Net) connector and your TE to the TE (Usr) connector. Activate the line from LT or TE.

6.5.2 Auxiliary connectors

6.5.2.1 Power feeding

There are two connectors for 4 mm plugs white (+) and black (-) (pos. 6) to sink/source the power feeding. According to the ZVEI specification for L_{po} lines the recommended voltage for feeding an TE is 48 VDC (± 9 VDC). The red LED (pos. 7) indicates a feeding on the line. The use of this connector is different in the three modes.

TE emulation: If a voltage for feeding the TE is generated by the LT, the voltage can be shunted and/or accessed by a voltmeter.

LT emulation: You can feed your TE by connecting to an external source (both versions) or internal source (version 2 only).

Monitor: If a voltage for feeding the TE is generated by the LT, the voltage is routed through to the TE and can be shunted and/or accessed by a voltmeter.

To prevent a possible damage of the the transformers in case another power source is on the line the maximum current is limited to 63 mA by an internal fuse in the connector module. If the fuse is blown it can be easily replaced (see view of connector module for location).

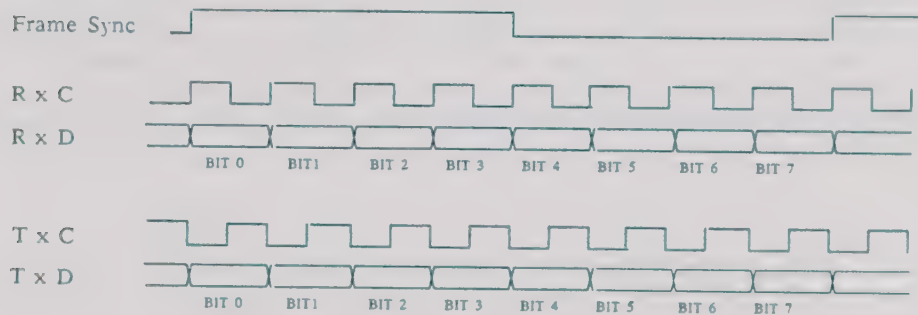
6.5.2.2 External B channel access

The B channels can be accessed by a female 15-pin Cannon connector (pos. 5).
There are two types of data:

- 1.) 64 kBit/s data streams with clocks, TTL level respective to pin 15
- 2.) Voice audio signals for a handset:
Symmetric, sensivity (programmable) -20 dB, bandwidth 200 - 3500 Hz,

Pin 1:	Output R x D / B1 from Net
Pin 2:	Ground (version 1), Output (+) voice audio (version 2)
Pin 3:	Output R x D / B2 from Usr
Pin 4:	Input T x D / B2 (use only in emulation mode)
Pin 5:	Ground (version 1), Input (+) voice audio (version 2)
Pin 6:	Output R x C (64 kHz)
Pin 7:	Output Frame Sync B1 (8 kHz)
Pin 8:	Output Frame Sync B2 (8 kHz)
Pin 9:	Output R x D / B2 from Net
Pin 10:	Output R x D / B1 from Usr
Pin 11:	Ground (version 1), Output (-) voice audio (version 2)
Pin 12:	Input T x D / B1 (use only in emulation mode)
Pin 13:	Output T x C (64 kHz, inverted to R x C)
Pin 14:	Ground (version 1), Input (-) voice audio (version 2)
Pin 15:	Ground

Timing diagrams



6.6 Principles of transmission

The ISDN-U_{po} connector module includes two IBC-Chips providing full duplex two-wire data transmission of the 144 kBit/s user bit rate via twisted pair telephone cables with up to 26 dB insertion loss at the Nyquist frequency of 192 kHz. Applying the time compression multiplex (TCM) technique the circuit requires a line bit rate 384 kBit/s, made up of twice the 144 kBit/s rate of the 2B + D channels and a 8 kBit/s rate for synchronization and maintenance. The B channels are scrambled while the D channel remains transparent. Taking into account a guard time of 2 bits, the remaining cable group delay amounts to about 23 microseconds for either direction; for example, highly capacitive 0.6 mm PVC cables up to 1.7 km limited by insertion loss and 0.6 mm PE cables up to 3.9 km limited by group delay can be equalized by the internal circuitry. For a 0.6 mm cable the DC feeding range is at least 4.7 km.

6.7 Hardware description

The ISDN-U_{po} connector module has two identical line interfaces. One is dedicated for the connection as a LT (X1), the other for the connection as a TE (X2). Both include overvoltage protectors (VD1, VD2, V1 to V8), symmetry coils (L1,L2) and the line transformers (T1,T2). Two IBC-Chips (D2, D3, PEB 2095, SIEMENS) provide a conversion from the line interface to the IOM-Interface. The clock oscillator (G1, version 1 only) generates the master clock for the IBCs and their internal PLLs. The tristate buffer 74HCT245 (D4) switches the data and frame clocks from the external to the internal source. The EPLD EP 600 (D1, EPLD = Erasable Programmable Logic Device) is used for the data routing of the two B channels and the D channel in monitor mode. The feeding voltage is limited by a VDR resistor (R11) and rectified by four diodes (V9 to V12). The optocoupler (D5) decouples the signal "FEEDING" from the line and the red LED indicates when a feeding is present. The connectors for external feeding (X4) are current protected by a fuse (F1). The signals for the B-channel access are only routed from the interface board to the connector (X5). The jumper positions (J1,J2) need to be jumpered in version 2.

7 LITERATURE

The following list displays specifications or publications used in this documentation.

Titel	Verfasser
ISDN BAI2 INTERFACE Baugruppenbeschreibung	Th. Hintze SIEMENS AG-MWB E311A
KURSORDNER TELECOM-S ₀	SIEMENS AG SCHULE FÜR MIKROELEKTRONIK
ISDN Das künftige Fernmeldenetz der Deutschen Bundespost	P. Kahl
THE VME-BUS SPECIFICATION	MOTOROLA

ISDN PRI INTERFACE

Protocol Tester K1197

SIEMENS AG

Revision Date	:	August, 1993
Engineering	:	AUT E33 D / Pa
Quality Control	:	MWB KQP 

1	INTRODUCTION	1-1
1.1	INTERFACE VERSIONS	1-1
1.2	REPLACING THE INTERFACE BOARD	1-2
1.3	CHANGING THE BASE ADDRESS OF THE PRI	1-2
2	FUNDAMENTAL CONSIDERATIONS	2-1
2.1	PRIMARY RATE ACCESS	2-1
2.2	T1-INTERFACE (1.544 MBIT/S)	2-2
2.3	S2M-INTERFACE (2.048 MBIT/S)	2-3
3	TEST SETUPS	3-1
3.1	MONITOR	3-2
3.2	EMULATION	3-3
3.3	DROP AND INSERT	3-4
4	INTERFACE BOARD	4-1
4.1	DESCRIPTION	4-2
4.2	BLOCK DIAGRAM	4-3
4.3	TECHNICAL DATA	4-4
4.4	MEANING OF LEDS	4-4
5	CONNECTOR MODULE T1 / S2M	5-1
5.1	DESCRIPTION	5-2
5.2	PIN ASSIGNMENTS MONITOR	5-2
5.3	PIN ASSIGNMENTS SIMULATION / EMULATION	5-3
5.4	PIN ASSIGNMENTS DROP AND INSERT	5-4
5.5	PIN ASSIGNMENTS 64 KBIT/S	5-5
5.6	TECHNICAL DATA	5-6
5.7	CONNECTOR MODULE T1	5-6
5.7.1	RECEIVER T1	5-6
5.7.2	TRANSMITTER T1	5-7
5.8	CONNECTOR MODULE S2M	5-11
5.8.1	RECEIVER S2M	5-11
5.8.2	TRANSMITTER S2M	5-12
6	PREAMPLIFIERS	6-1
6.1	PROTECTED MONITORING POINT PMP	6-1
6.2	CONNECTOR MODULES WITH PREAMPLIFIERS	6-1
7	APPLICABLE DOCUMENTS	7-1

1 INTRODUCTION

This document describes the function of the ISDN-Primary-Rate-Interface (PRI) in the SIEMENS K1197 protocol tester.

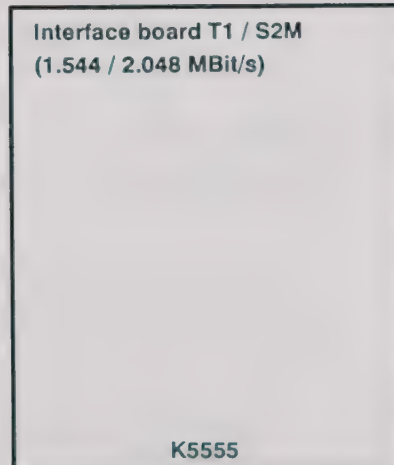
1.1 INTERFACE VERSIONS

There are two basic versions of the PRI :

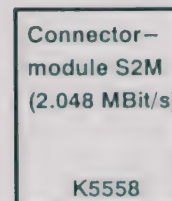
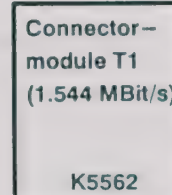
- Version 1 is used for transmission at **1.544 Mbit/s**. The main countries of applications are the USA, Japan and Canada. This interface is referred to as the **T1-Interface** in the following text.
- Version 2 is used for transmission with **2.048 Mbit/s**. This **CEPT-Interface** or **S2M-Interface** is mainly used in Europe, but also e.g. in Australia.

Since the actual **Interface board** is designed for both versions and can be switched to the respective transmission rate using the software, the only difference concerns the external **Connector module**.

This is therefore available as the **Connector module T1** and as the **Connector module S2M**.



Within the K1197



Externally connected option

1.2 REPLACING THE INTERFACE BOARD

When handling the board, the regulations for electrostatically sensitive devices (ESD) must be observed!

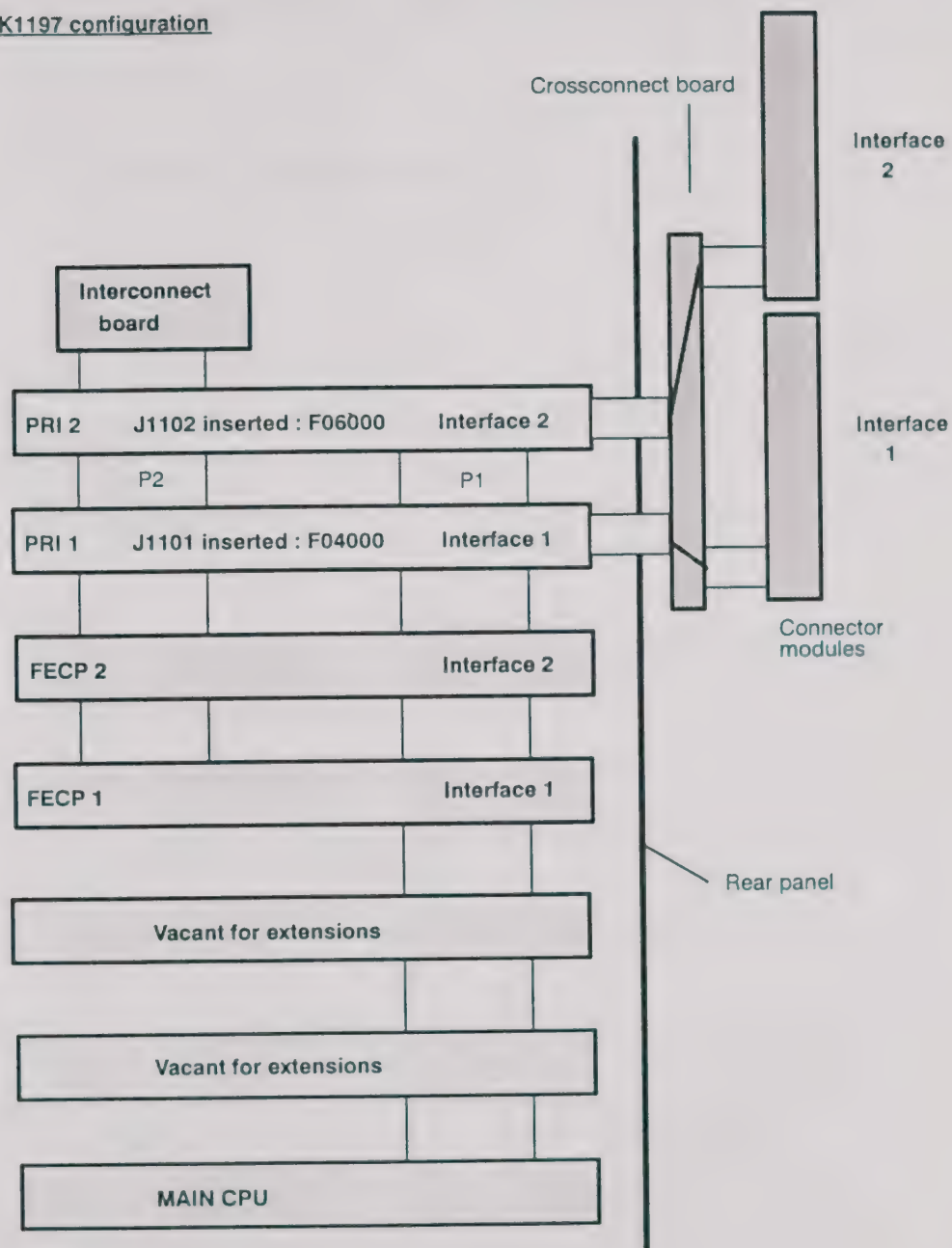
Proceed as follows to replace the interface board :

1. Switch off K1197
2. **Disconnect mains plug from device**
3. Remove connector module(s) and crossconnect board
4. Remove housing cover
5. Pull out interface board
6. Insert new interface board
7. Screw back housing cover
8. Reattach crossconnect board and connector module(s)
9. Reconnect mains plug and switch device on

1.3 CHANGING THE BASE ADDRESS OF THE PRI

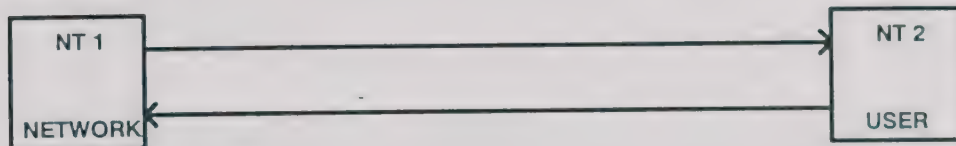


Changing the address on –K5555–B701.

K1197 configuration

2 FUNDAMENTAL CONSIDERATIONS

2.1 PRIMARY RATE ACCESS



point-to-point configuration

NT 1 : Network termination, NETWORK

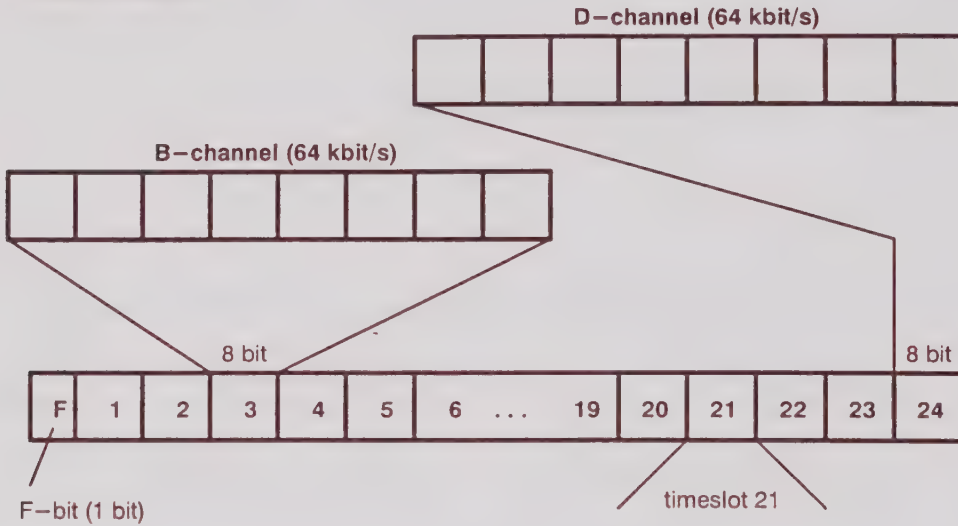
NT 2 : e.g private branch exchange, USER

- The interface is always operated as a point-to-point configuration, i.e. such that
 - referred to level 1 -- a data source (transmitter) and a data sink (receiver) are connected together via the interface.
- The transmission is via 4 conductors with symmetrical pairs.
- The **USER (NT 2)** synchronizes the data to be transmitted with the clock received from the **NETWORK (NT 1)**.
- Two bit rates have been defined by CCITT for the primary rate access :

T1 :	CEPT :
-- 1.544 Mbit/s (24 Channels)	-- 2.048 Mbit/s (31 Channels)
23 B at 64 kbit/s	30 B at 64 kbit/s
1 D at 64 kbit/s	1 D at 64 kbit/s
or	or
24 B at 64 kbit/s	31 B at 64 kbit/s

2.2 T1–INTERFACE (1.544 MBIT/S)

Frame structure



F-bit : used for Cyclic Redundancy Check (CRC 6 according to CCIT T Rec. G.704)

Code : AMI, B8ZS

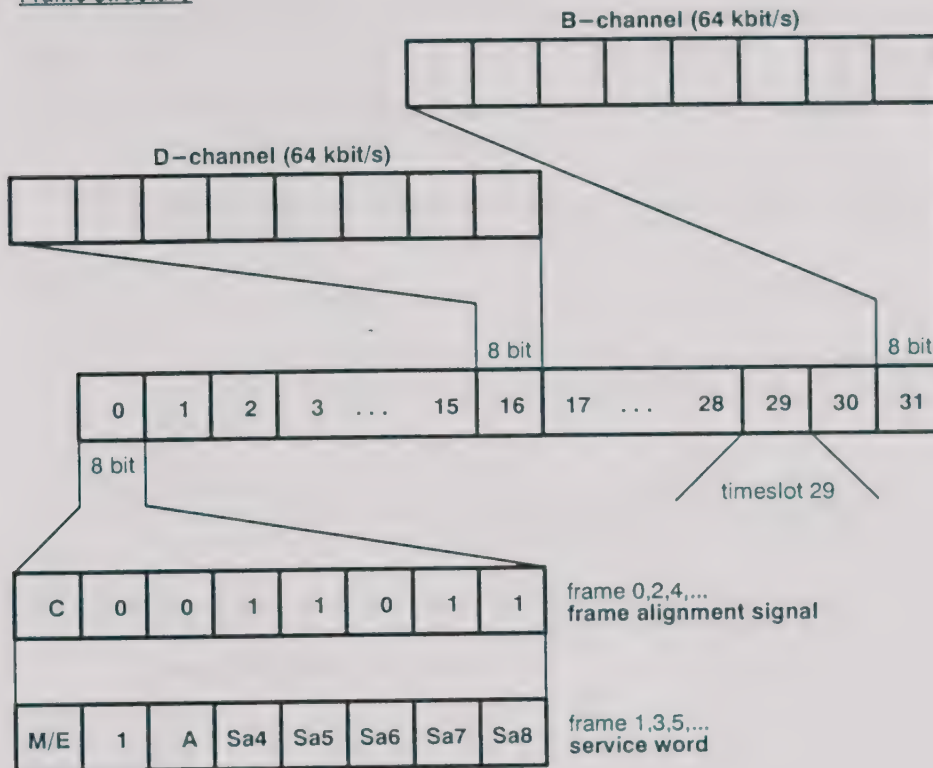
Frame length : 193 bit, 1...193

Organization : F-bit (bit 1) + 24 timeslots (8 bit each), 1...24

Framing frequency : 8 kHz

2.3 S2M-INTERFACE (2.048 MBIT/S)

Frame structure



C : Cyclic Redundancy Check (CRC 4) bits

M : Multiframepattern (001011)

E : CRC 4 Error indication bit

A : Remote alarm indication

Sa5 ... Sa7 : for national use

Sa4, Sa8 : for international use

Code : HDB3

Frame length : 256 bit, 1...256

Organization : 32 timeslots (8 bit each), 0...31

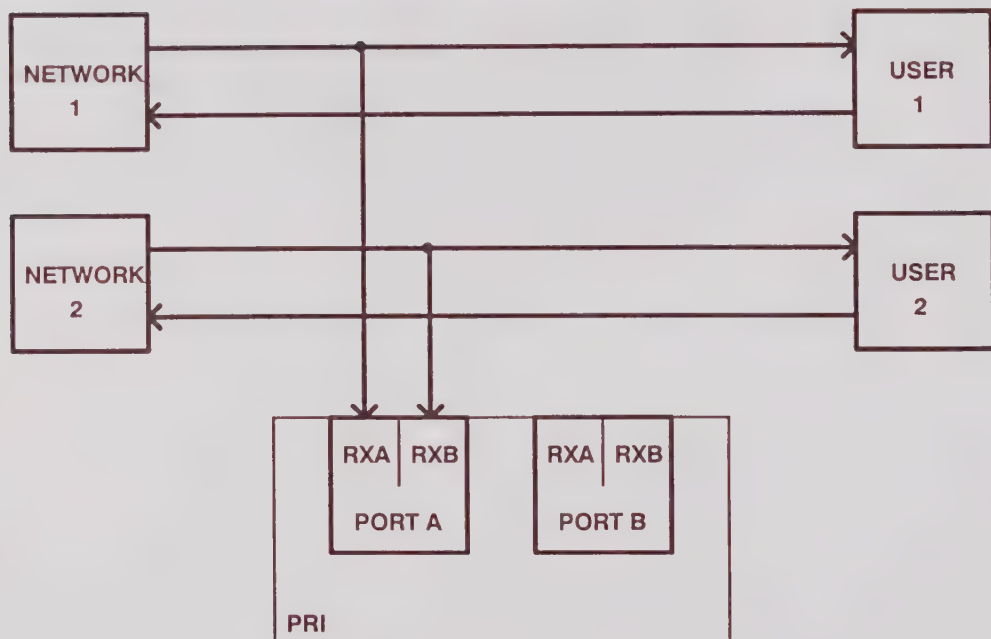
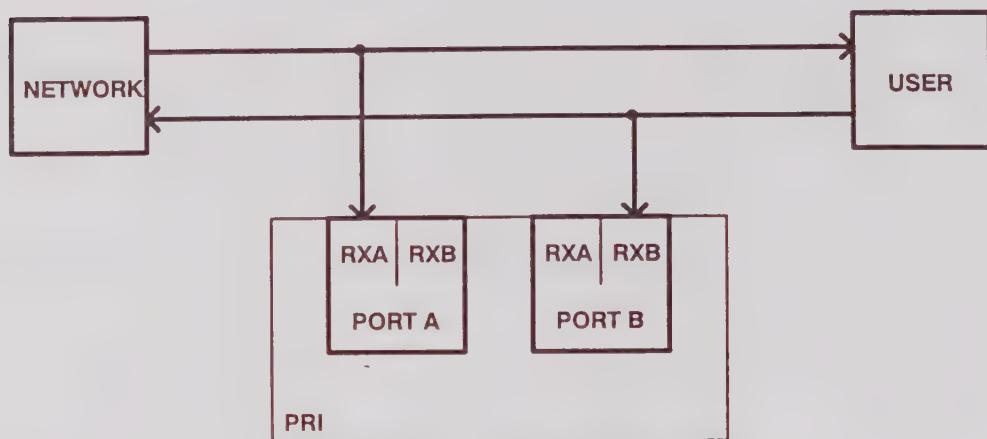
Framing frequency : 8 kHz

3 TEST SETUPS

The following module characteristics must be known in order to understand the following test setups:

1. A **PRI** in the SIEMENS protocol tester has **2** independent connections to a PCM line : **PORT A** and **PORT B**.
2. Each port can be configured **independent** of the other one as a **USER (NT 2)** or **NETWORK (NT 1)**.
3. Each of these ports normally consists of **one** transmitter and **one** receiver.
4. The received data of each port can be processed and monitored in the protocol tester or also transmitted again by the **attached transmitter** (e.g. RXA → TXA) of this port.
5. In **MONITOR** mode the transmitters are switched off and are replaced by receivers of each other port. So it is possible to connect both directions of a PCM link to **PORT A** or **PORT B** or one direction to **PORT A** and one direction to **PORT B**.
6. In **DROP AND INSERT** mode – as a difference to **SIMULATION / EMULATION** – the transmitters **TXA** and **TXB** are swapped at the connector. The advantage is the easier connecting of the cables.

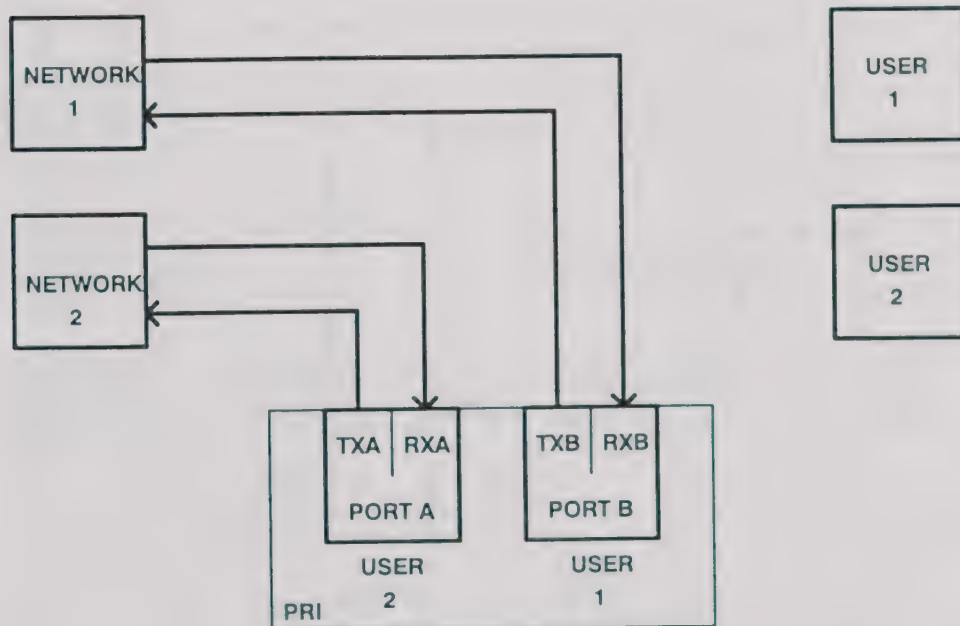
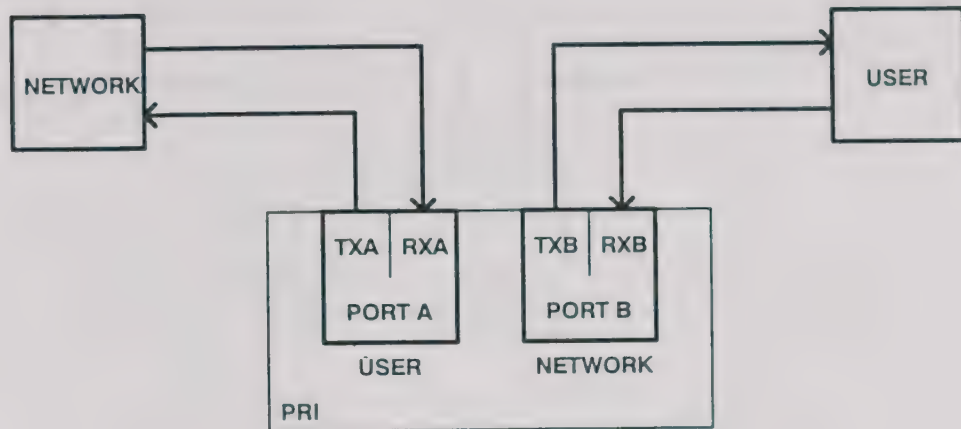
3.1 MONITOR



In this mode data passes between an active NETWORK and USER and is not disturbed by the tester. Each receiver is connected to one direction of the link.

Selected channels may be demultiplexed and monitored.

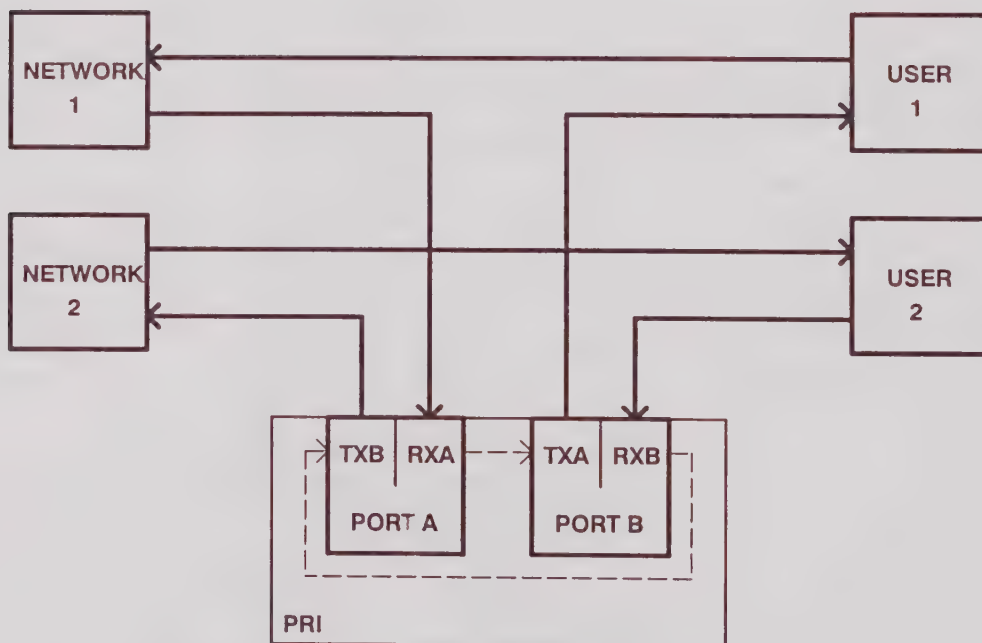
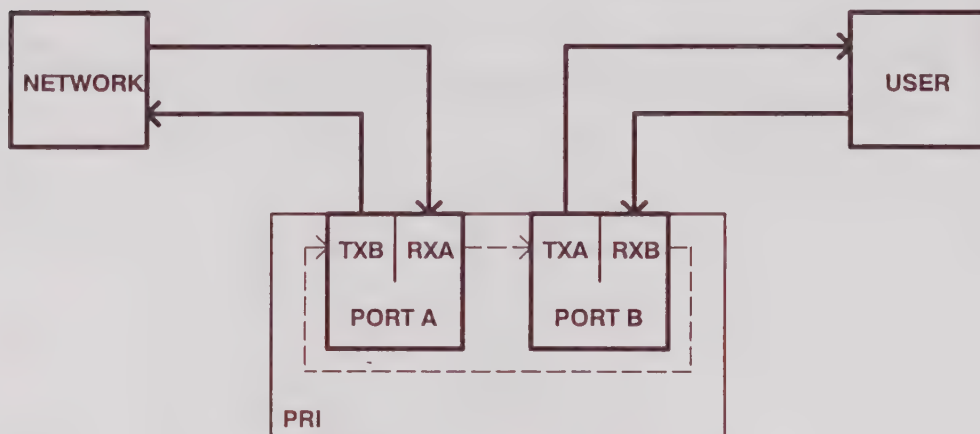
3.2 EMULATION



Each primary rate transceiver can emulate a USER or NETWORK and is independent of each other with regard to timing and channel selection etc.

In this operating mode the protocol tester is a true **dual port machine**.

3.3 DROP AND INSERT

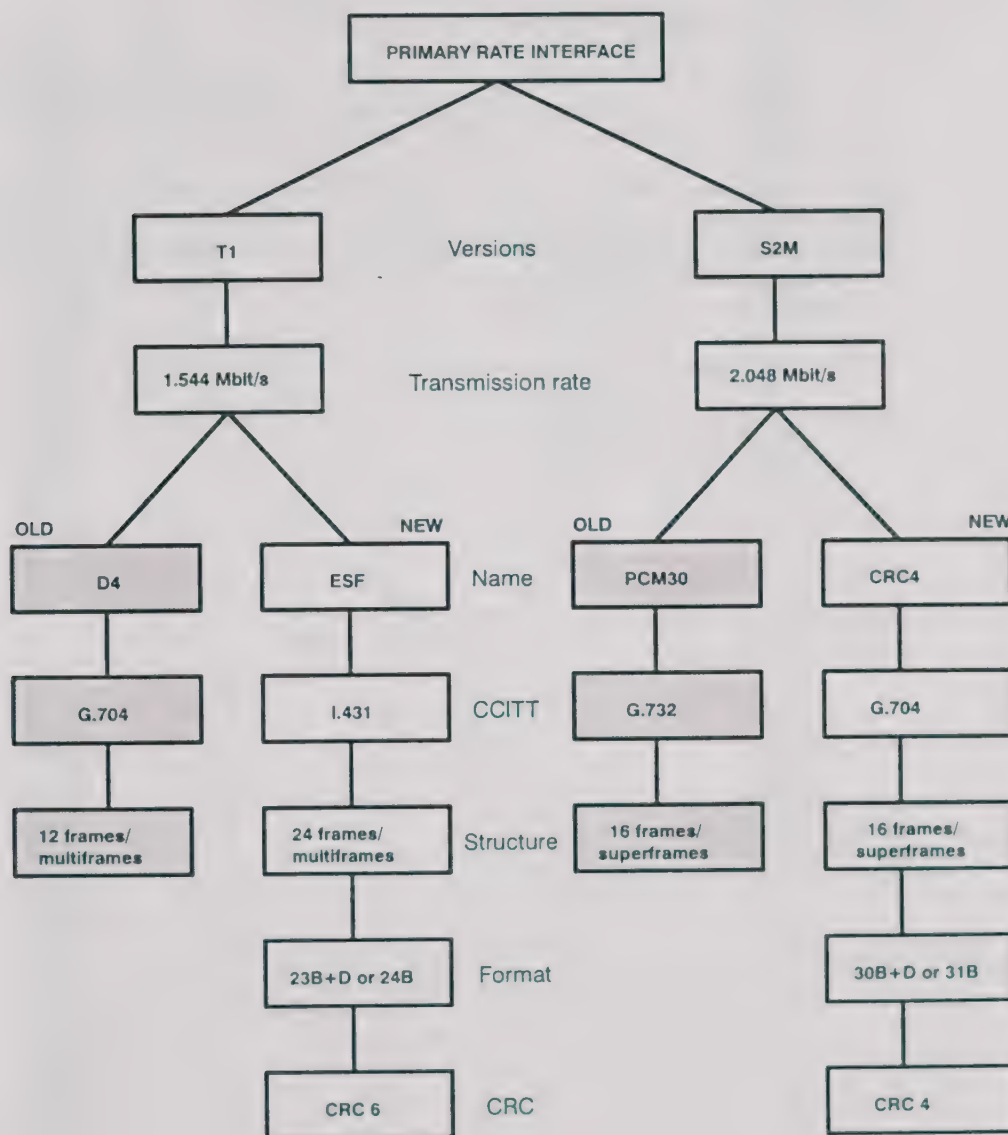


In this mode each PORT handles one direction of the primary rate link from a NETWORK to a USER.

Selected channels may be dropped in both directions, data inserted or can be monitored..

Unaffected channels may pass through transparently (RXA → TXA and RXB → TXB).

4 INTERFACE BOARD



4.1 DESCRIPTION

The interface board is designed as a multilayer (8 layers) with a double Europa format. It does not have its own microprocessor and is configured via the VME bus (connector P1). Data transmission to the processing computer (FECP) is via connector P2.

The PRI is of largely symmetrical design to enable independent use of PORT A and PORT B.

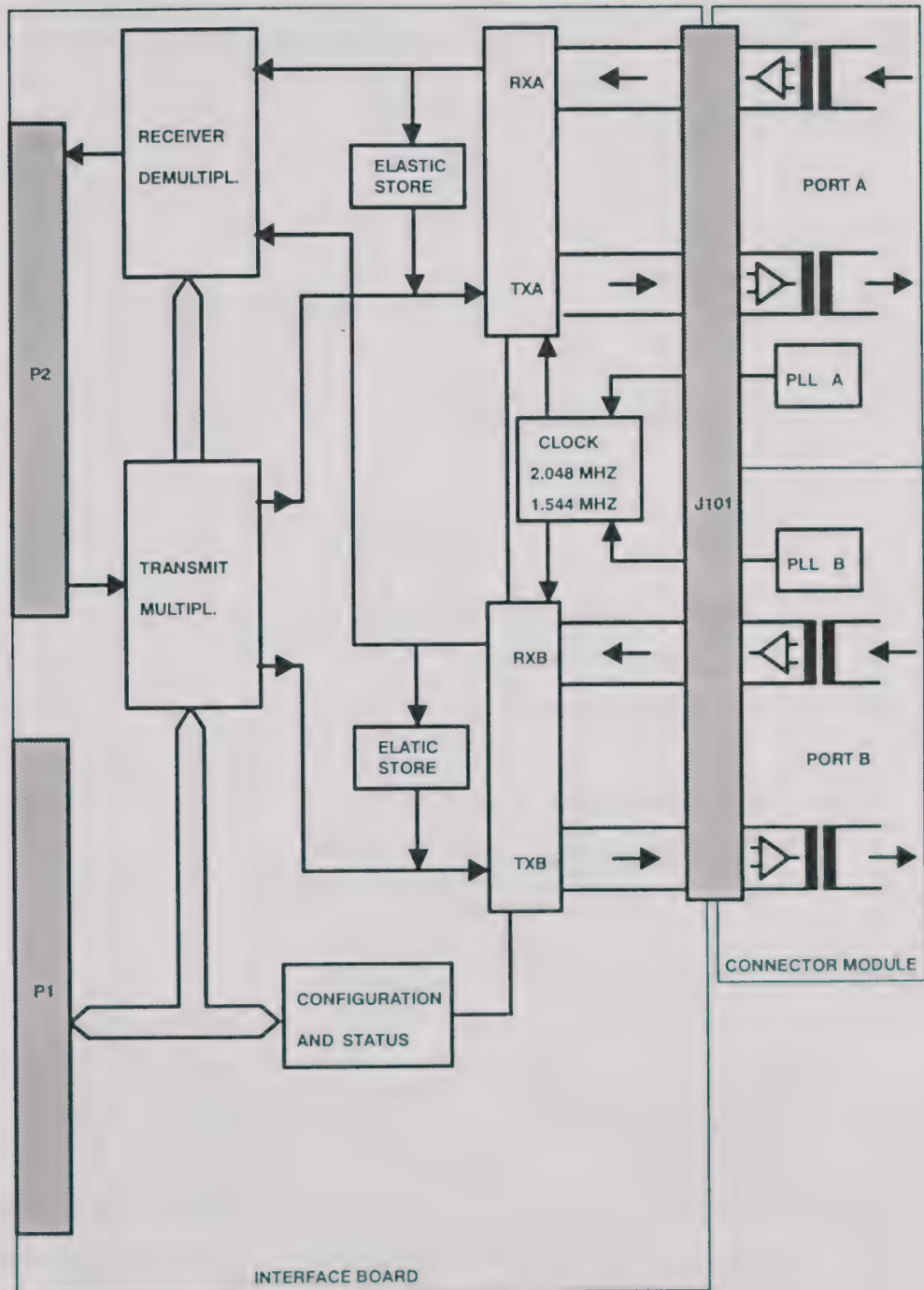
The interface to the programmer consists of a register bank in which the transmission-specific settings can be carried out of layer 1 such as the operating mode (D4, ESF, PCM30 or CRC4), test setup (monitor, emulation or drop and insert), terminating resistance etc.

In addition a control register is present for each timeslot in the frame of the transmitter and receiver in order to access the transmitter multiplexers and receiver demultiplexers. A timeslot is transmitted e.g. to the processing computer within a time which is shorter than the time required to receive the following timeslot. Thus any selected timeslot combinations can be transmitted as a logical channel between PRI and FECP.

This possibility exists completely independently for PORT A and PORT B.

The PRI thus already supports frame formats such as the **H0-Channel** (6*B), **H11-Channel** (24*B) and **H12-Channel** (30*B).

4.2 BLOCK DIAGRAM



4.3 TECHNICAL DATA

Clock accuracy: 1.544 MHz $\pm$ 25 ppm

2.048 MHz $\pm$ 25 ppm

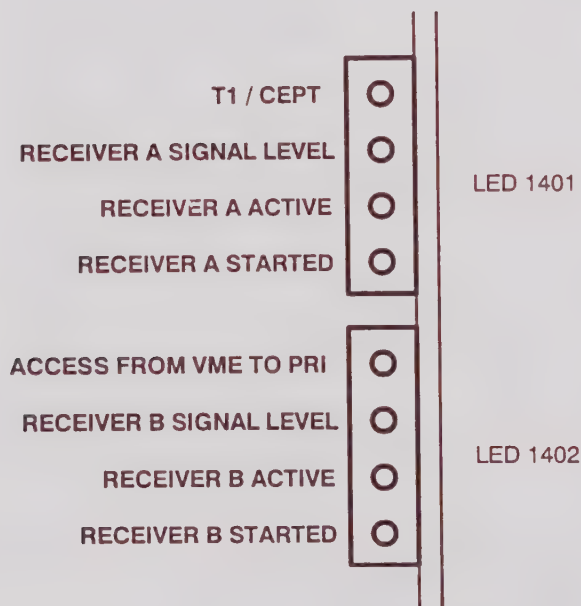
Current consumption: + 5V 1.70 A typ.

+12V 0.02 A typ.

−12V 0.02 A typ.

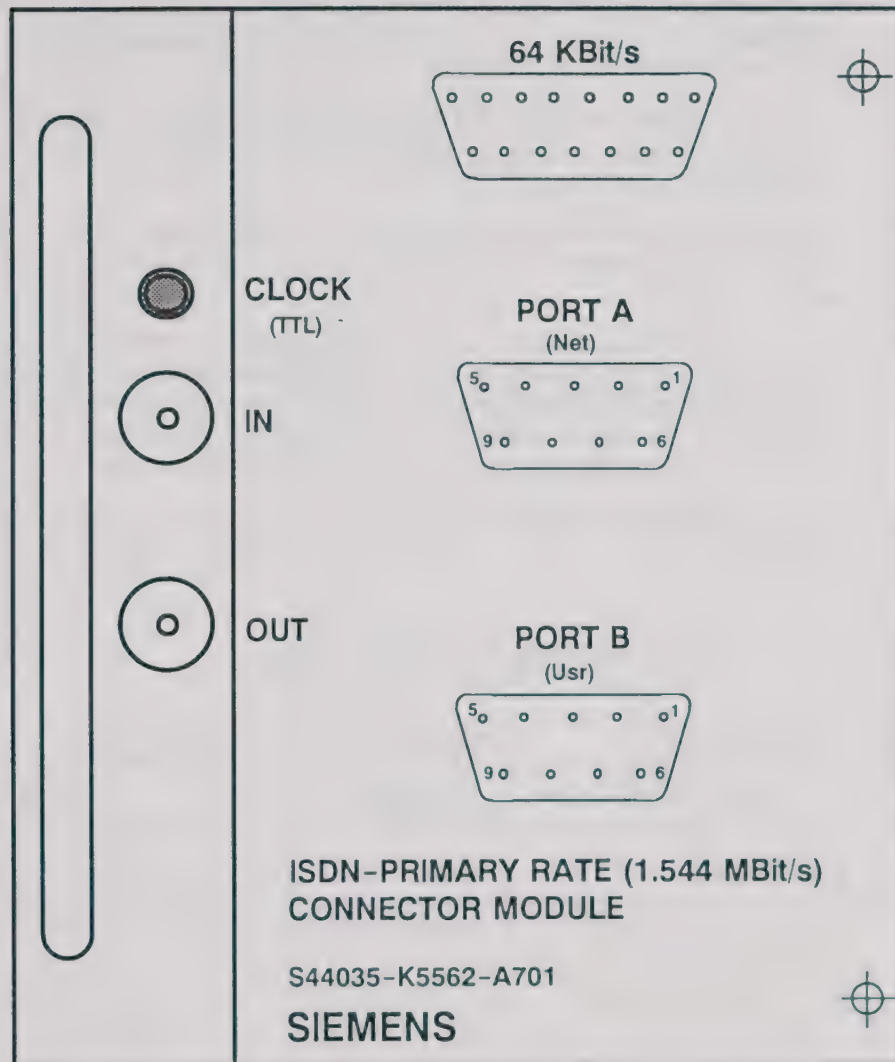
Temperature range : 0 – 70 °C

4.4 MEANING OF LEDS



LED	ON	OFF
T1 / CEPT	CEPT (2.048 MHz)	T1 (1.544 MHz)
SIGNAL LEVEL	OK	no level or too low
ACTIVE	receiver synchronized	receiver not synchronized
STARTED	start bit set	no start bit set
ACCESS VME TO PRI	access by VME bus	no access

5 CONNECTOR MODULE T1 / S2M



DIP switch S1501 and S1502 :



Front is down (ON)



Front is up (OFF)

5.1 DESCRIPTION

The connector module is the actual connection to the PCM link. Repeating coils isolate the electronics from the line. The defined pulse shapes and transmitter levels are generated at the transmitter end. The receiver end consists of a clock recovery system in order to enable synchronisation of the USER with the NETWORK. The received signals are also converted into TTL signals for further processing.

The DB 9 pin female connectors **PORT A** and **PORT B** are used for the connection to the PCM link. For test purposes the pin assignments can be changed using the DIP switches S1501 and S1502 (accessible from the side).

The DB 15 pin female connector **64 KBit/s** allows external access to any B-channel. Data, clocks and frame information are provided by the connector module for this purpose as **TTL signals**.

An external clock (TTL level) with 1.544 MHz (K5562) or 2.048 MHz (K5558) can be applied to the BNC socket **CLOCK IN**. The internal crystal clock is automatically switched off as soon as the external clock is applied, and the PRI then operates with this external frequency. The LED additionally lights up on the connector module to indicate this mode. It is thus possible to use a clock coupled to other devices or to use a highly accurate reference clock.

The socket **CLOCK OUT** outputs the clock used on the module (1.544 / 2.048 MHz) as a TTL signal.

5.2 PIN ASSIGNMENTS MONITOR

PORT A = PORT B

(CONNECTION TO THE PCM LINE)

PIN	SIGNAL NAME	FUNCTION	DIRECTION
9	RXB+	receive B positive	Input
5	RXB-	receive B negative	Input
6	RXA+	receive A positive	Input
1	RXA-	receive A negative	Input
2,3,4	GND	earth/system ground	— — —

In MONITOR mode the transmitters are switched off and are replaced by receivers of each other PORT.

5.3 PIN ASSIGNMENTS SIMULATION / EMULATION

PORT A

(CONNECTION TO THE PCM LINE)

PIN	SIGNAL NAME	FUNCTION	DIRECTION
9	TXA+	transmit A positive	Output
5	TXA-	transmit A negative	Output
6	RXA+	receive A positive	Input
1	RXA-	receive A negative	Input
2,3,4	GND	earth/system ground	- - -

PORT B

(CONNECTION TO THE PCM LINE)

PIN	SIGNAL NAME	FUNCTION	DIRECTION
6	TXB+	transmit B positive	Output
1	TXB-	transmit B negative	Output
9	RXB+	receive B positive	Input
5	RXB-	receive B negative	Input
2,3,4	GND	earth/system ground	- - -

5.4 PIN ASSIGNMENTS DROP AND INSERT

PORT A

(CONNECTION TO THE PCM LINE)

PIN	SIGNAL NAME	FUNCTION	DIRECTION
9	TXB+	transmit B positive	Output
5	TXB-	transmit B negative	Output
6	RXA+	receive A positive	Input
1	RXA-	receive A negative	Input
2,3,4	GND	earth/system ground	- - -

PORT B

(CONNECTION TO THE PCM LINE)

PIN	SIGNAL NAME	FUNCTION	DIRECTION
6	TXA+	transmit A positive	Output
1	TXA-	transmit A negative	Output
9	RXB+	receive B positive	Input
5	RXB-	receive B negative	Input
2,3,4	GND	earth/system ground	- - -

In DROP AND INSERT mode the transmitter of PORT A and the transmitter of PORT B are swapped.

5.5 PIN ASSIGNMENTS 64 KBIT/S

64 KBit/s CONNECTOR

(EXTERNAL CHANNEL ACCESS)

PIN	SIGNAL	FUNCTION	DIRECTION
6	RXDCK64EX	receive 64 kbps clock	Output
7	RXDFRMEX	receive 64kbps data frame clock	Output
8	TXDFRMEX	transmit 64 kbs data frame clock	Output
10	RXD64EX	receive 64 kbps data	Output
12	TXD64EX	transmit 64 kbps data	Input
13	TXDCK64EX	transmit 64 kbps clock	Output
1,2,3,4,5,9 11,14,15	GND	earth/system ground	— — —

5.6 TECHNICAL DATA

Current consumption : + 5V 0.10 A typ.

+12V 0.08 A typ.

−12V 0.02 A typ.

Temperature range : 0 – 70 °C

5.7 CONNECTOR MODULE T1

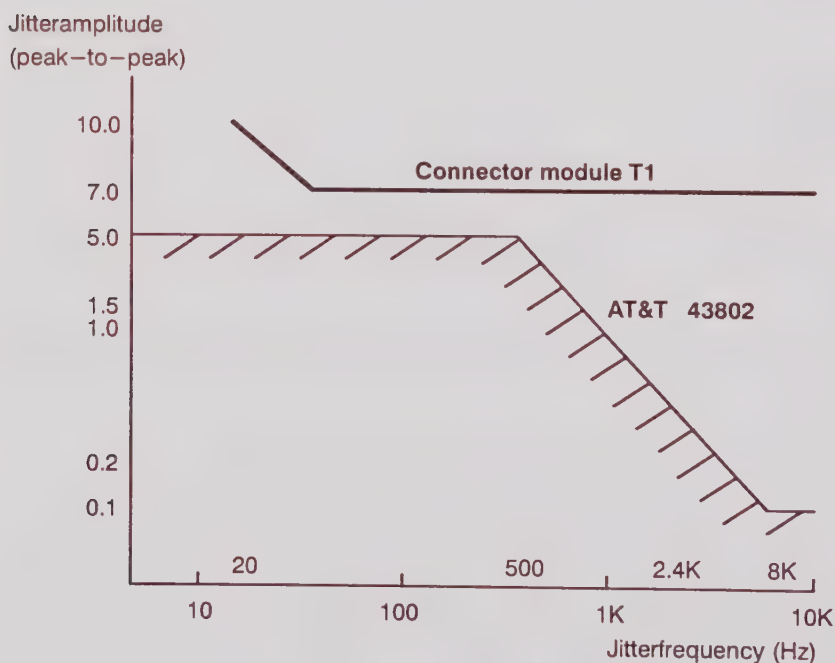
5.7.1 RECEIVER T1

Input impedance : 100 ohms (terminated)

(programmable) 1500 ohms (open)

Input voltage : 1 V_{pp} – 7 V_{pp}

Jitter :



Input Jitter Tolerance

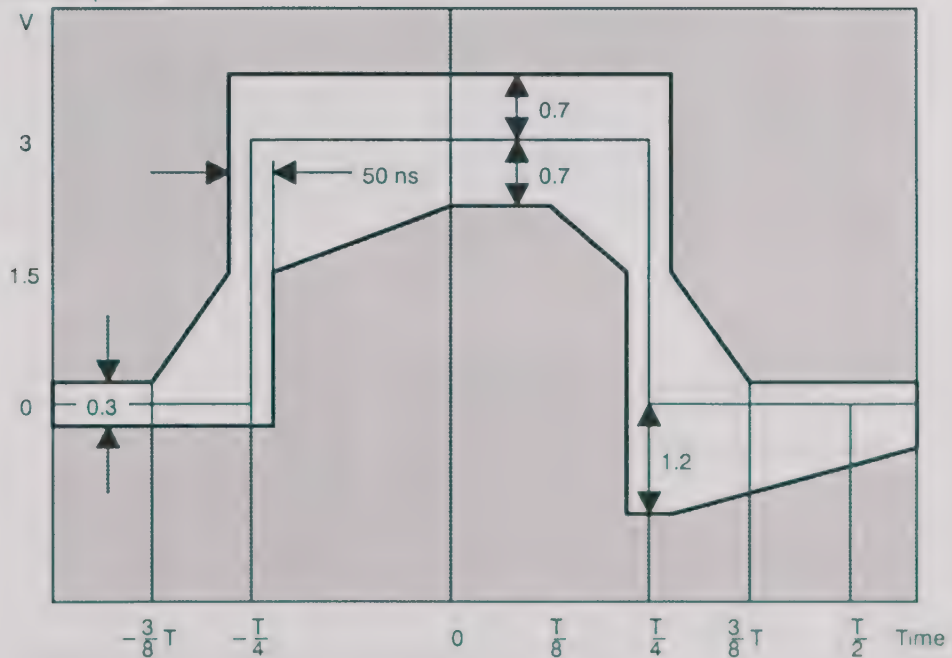
5.7.2 TRANSMITTER T1

Bit rate		1544 KBit/s
Pair(s) in each direction of transmission		One symmetric pair
Code		AMI or B8ZS
Test load impedance		100 ohms resistive
Nominal pulse shape		Rectangular
Signal-level	Power at 772 kHz	+12 dBm to +19 dBm
	Power at 1544 kHz	At least 25 dB below the power at 772 kHz

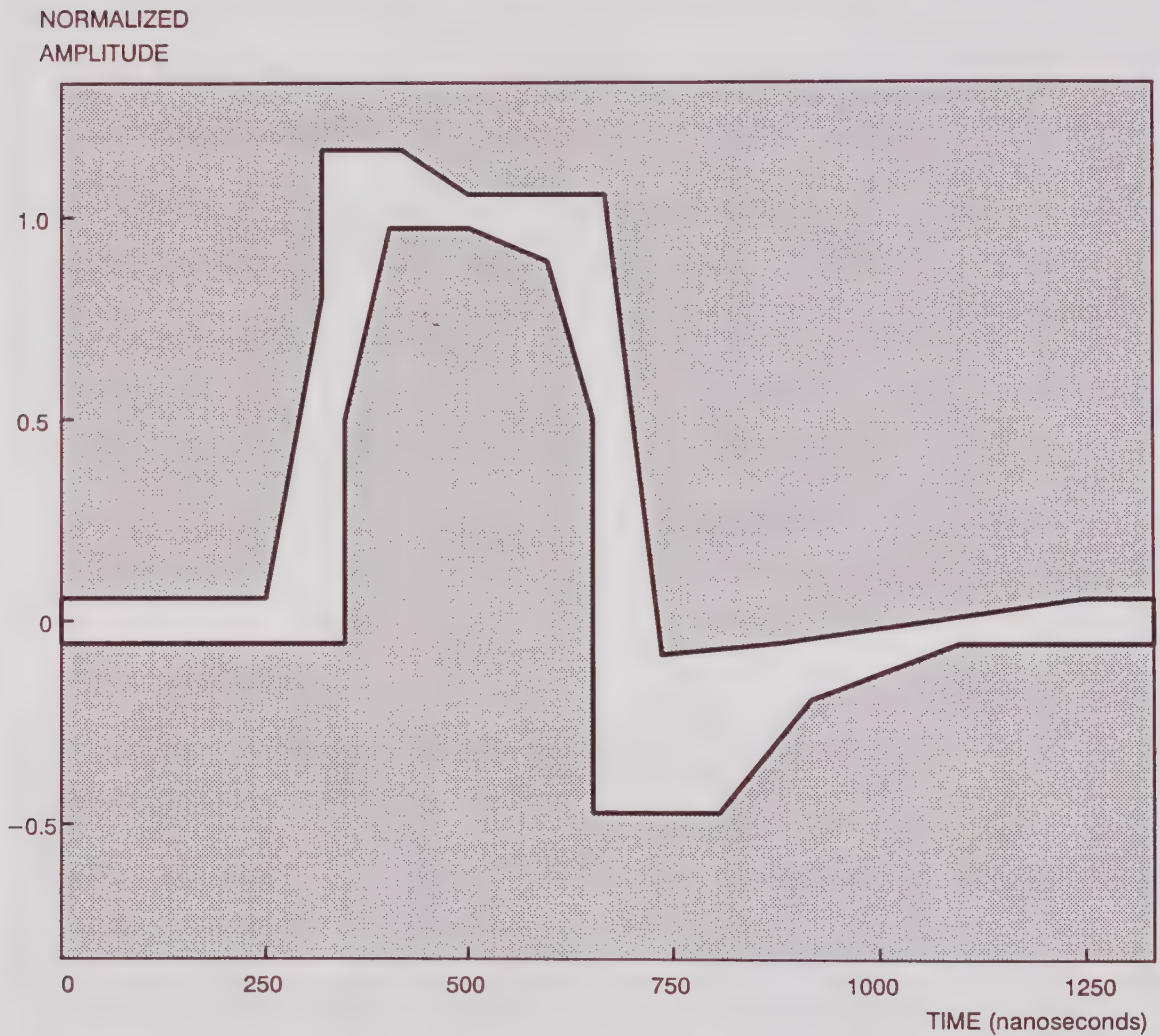
Specification at the Output

(CCITT G.703)

Pulse amplitude

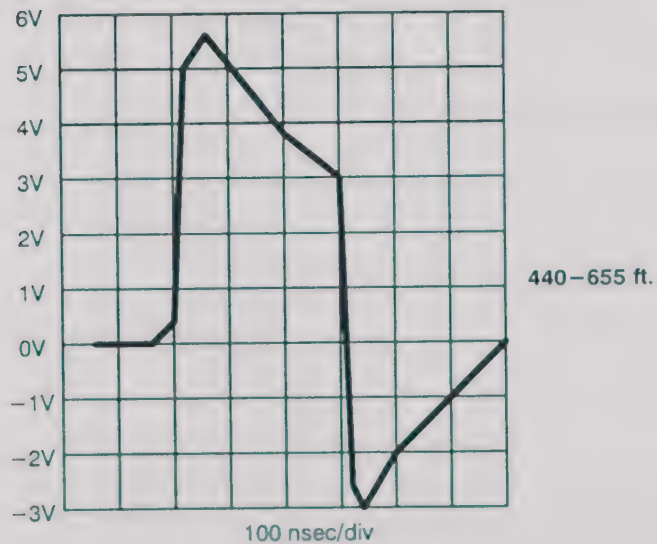
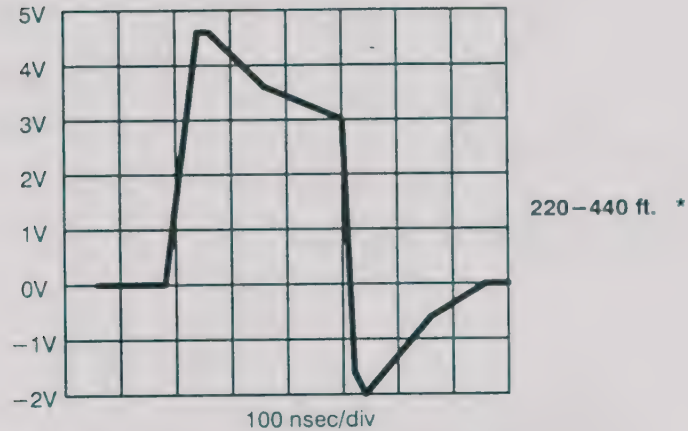
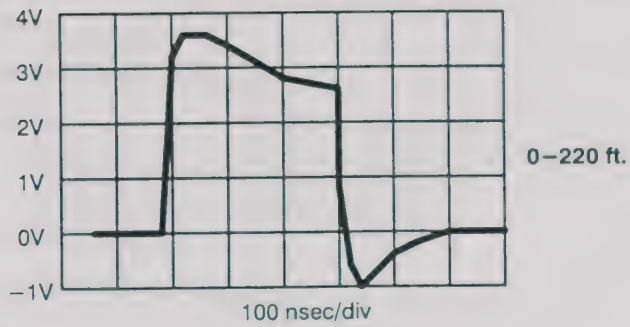
Pulse Mask for Interface at 1544 kbit/s

(CCITT G.703)



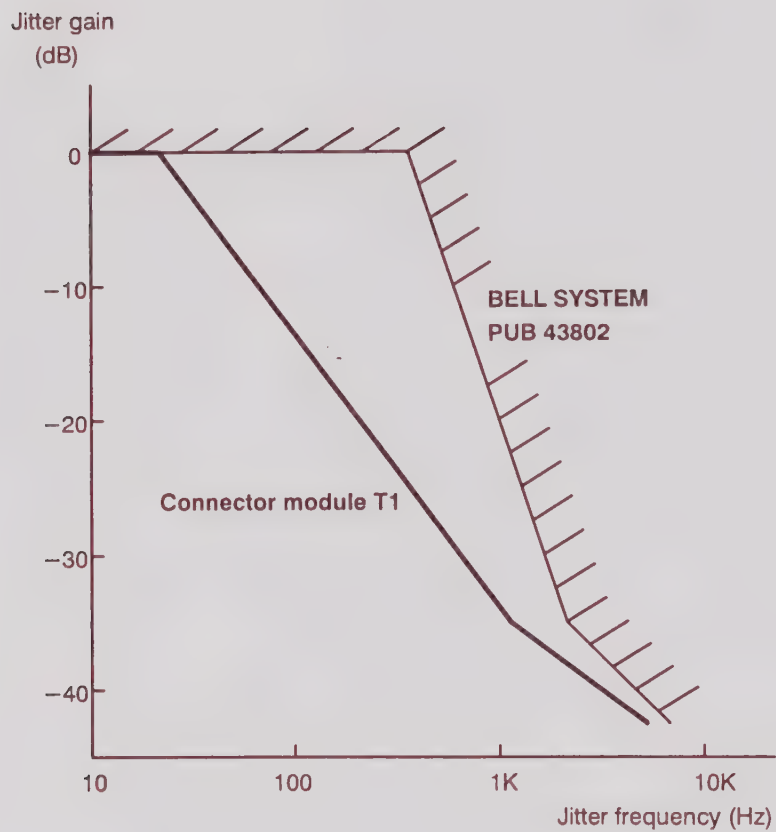
Mask of the Pulse at DSX-1 Cross Connect

(TECHNICAL ADVISORY # 34 , AT&T)



* not supported

Equalized Pulses for different Line Lengths



CCITT Jitter Attenuation Characteristics

5.8 CONNECTOR MODULE S2M

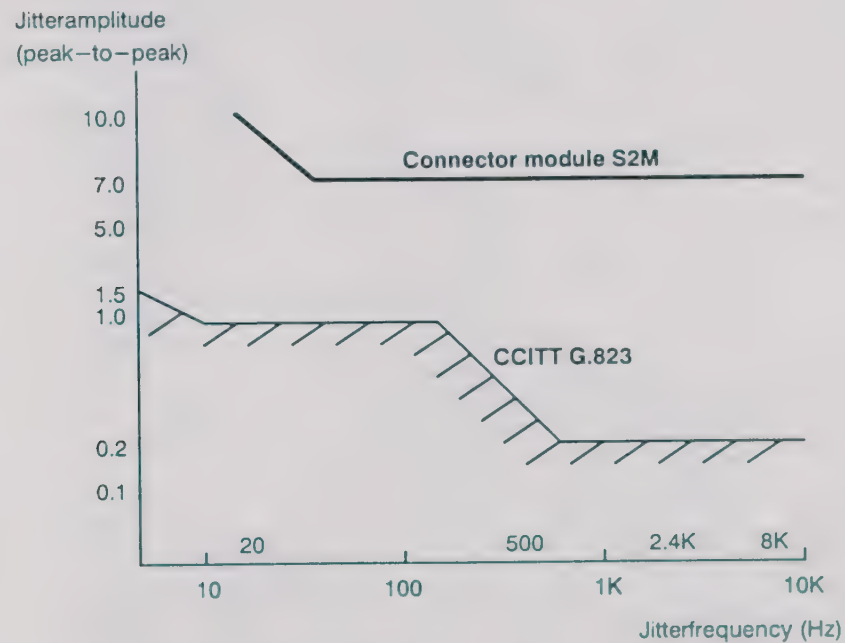
5.8.1 RECEIVER S2M

Input impedance : 120 ohms or 75 ohms (terminated)

(programmable) 1500 ohms (open)

Input voltage : 1 Vpp – 7 Vpp

Jitter :



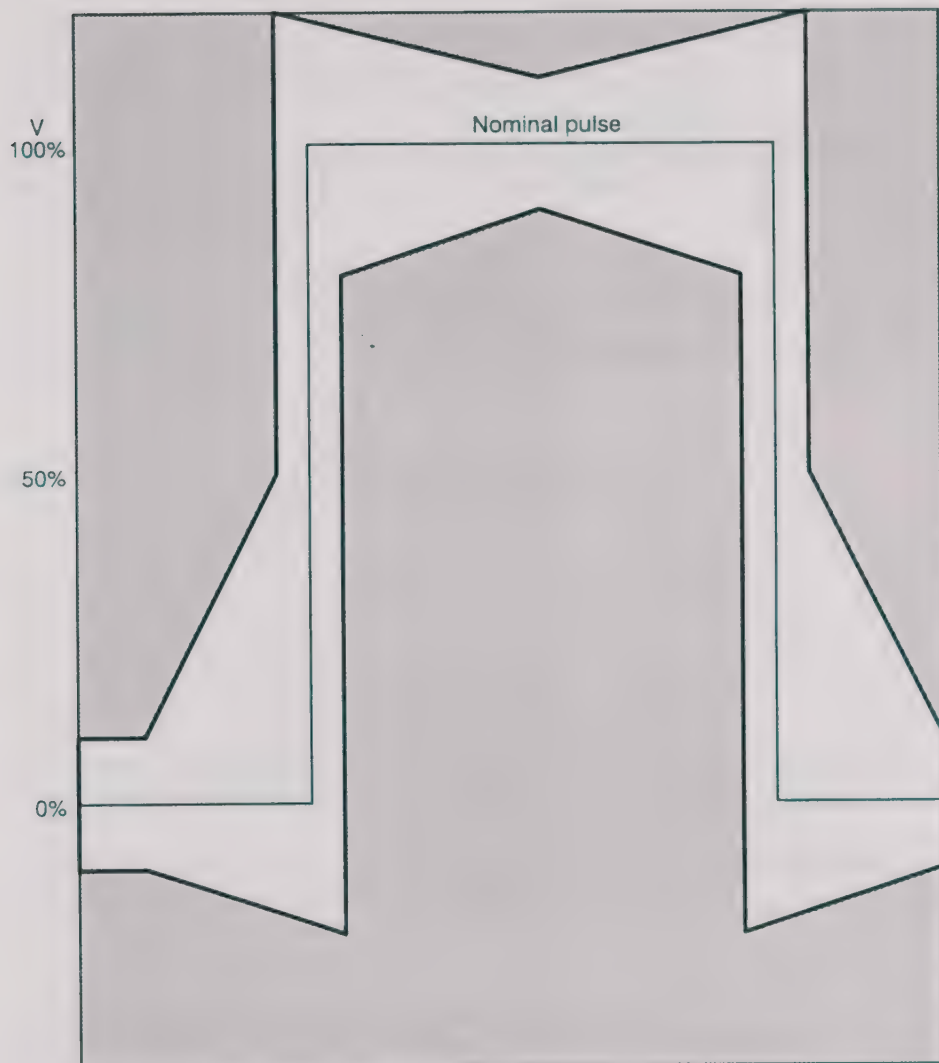
Input Jitter Tolerance

5.8.2 TRANSMITTER S2M

Pulse shape (nominally rectangular)	All marks of a valid signal must conform with the mask (Figure next page) irrespective of the sign. The value V corresponds to the nominal peak value.	
Pair(s) in each direction	One coaxial pair	One symmetrical pair
Test load impedance	75 ohms resistive	120 ohms resistive
Nominal peak voltage of a mark (pulse)	2.37 V	3 V
Peak voltage of a space (no pulse)	0 +/- 0.237 V	0 +/- 0.3 V
Nominal pulse width	244 ns	
Ratio of the amplitudes of positive and negative pulses at the centre of the pulse interval	0.95 to 1.05	
Ratio of the widths of positive and negative pulses at the nominal half amplitude	0.95 to 1.05	
Maximum peak-to-peak jitter at the output port	See CCITT G.823	

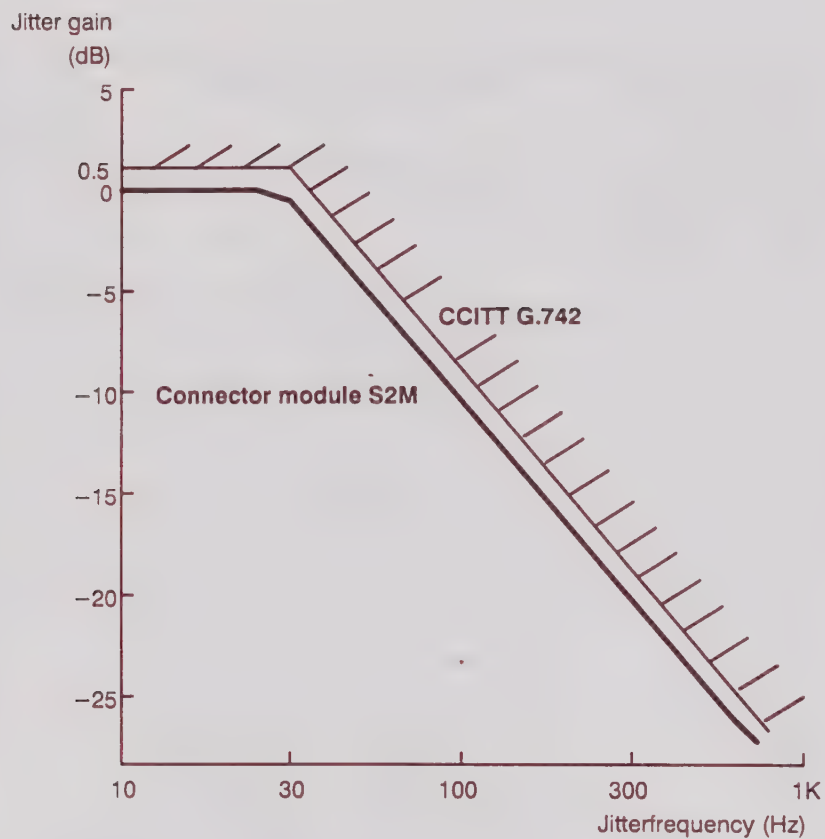
Specification at the output port

(CCITT G.703)



Pulse Mask for Interface at 2048 kbit/s

(CCITT G.703)

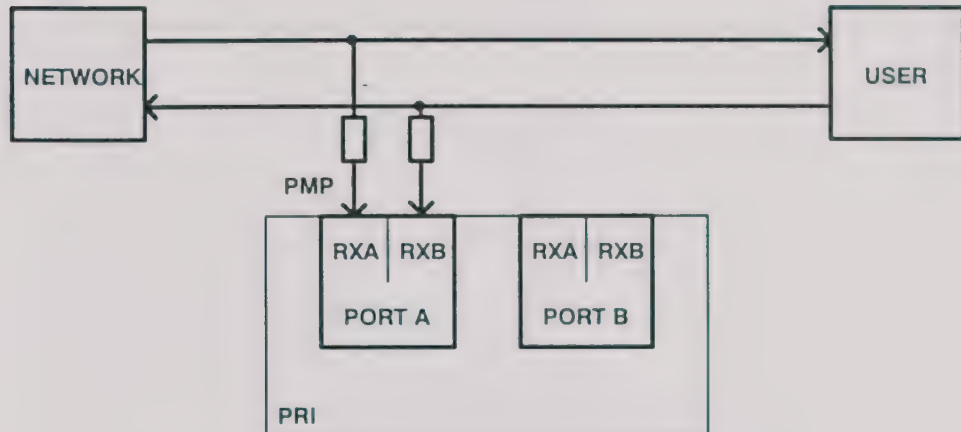


CCITT Jitter Attenuation Characteristics

6 PREAMPLIFIERS

6.1 PROTECTED MONITORING POINT PMP

CCITT defines in G.772 the **Protected Monitoring Point (PMP)**. A PMP provides a digital interface at which it is possible to monitor the transmitted signal and to make measurements with suitable test equipments.



6.2 CONNECTOR MODULES WITH PREAMPLIFIERS

Starting in 4th Quarter '93 the connector modules of the K1197 were delivered with an integrated preamplifiers to work on protected monitoring points. Modules with preamplifiers carry the label *High Performance Preamplifiers* on the face plate.

These new connector modules are compatible to the older ones.

The preamplifiers are automatically adapting to the input level.

The input sensitivity of the receiver with preamplifiers is significantly improved :

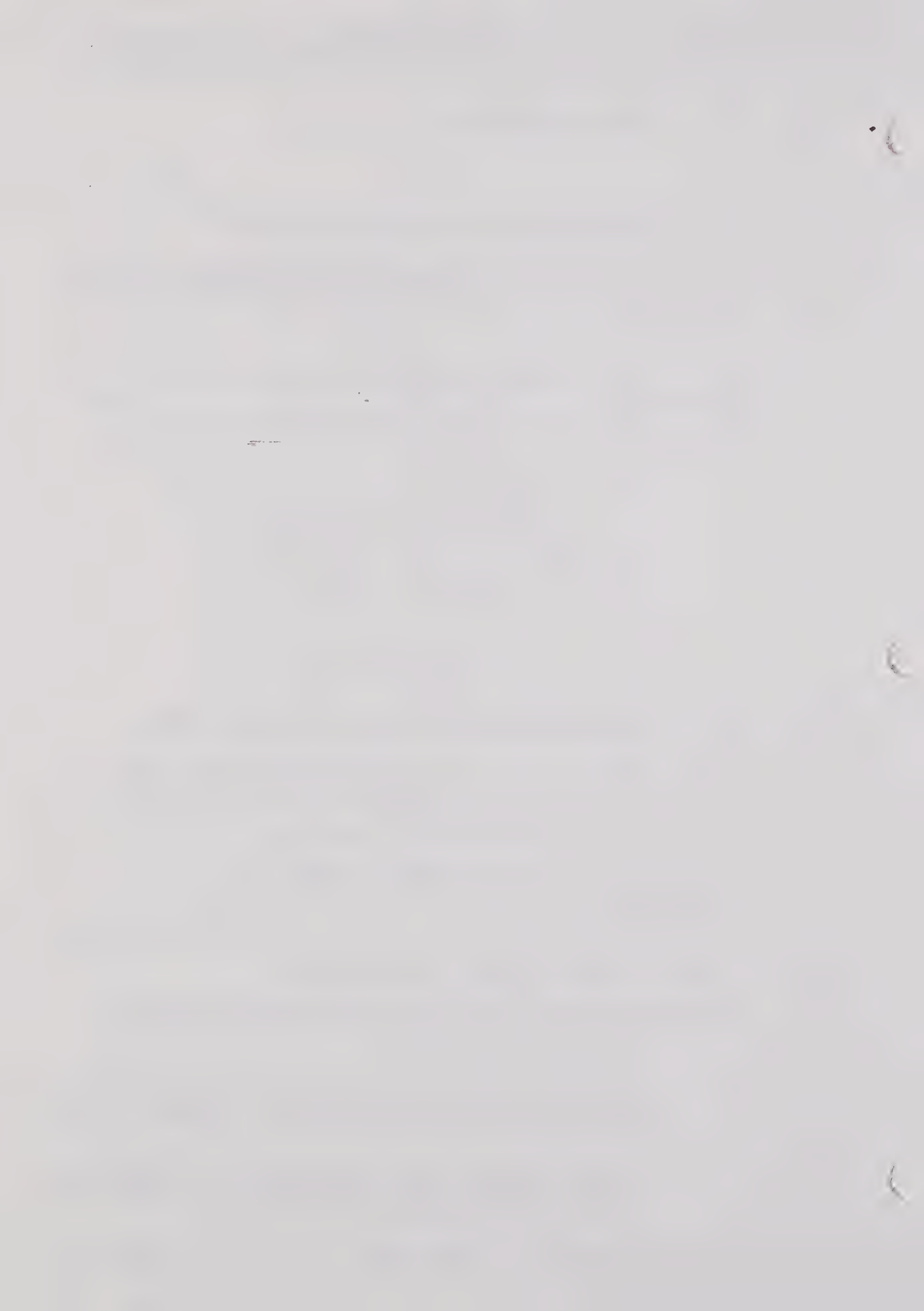
Input sensitivity : 0 to -37 dB

The receiver in MONITOR mode is now terminated with 75, 100 or 120 ohms. The high impedance to the line is guaranteed by the resistor in the **PMP**.

Important : New software versions will support the switching between *High Impedance* and *Low Impedance* (75, 100 or 120 ohms) in MONITOR/ANALYSER mode with an expansion in the *Setup Menu*.

As a temporary solution it is also possible to use *FORTH* commands for the switching of the termination resistor (75 ohms) **after finishing the normal setup** in the *Setup Menu* :

```
: R75_PRI1 0XF04201 C@ 0X01 OR 0XF04201 C! ; <-- (INTERFACE 1)
: R75_PRI2 0XF06201 C@ 0X01 OR 0XF06201 C! ; <-- (INTERFACE 2)
The activation is done using R75_PRI1 <-- or R75_PRI2 <-- .
```



7 APPLICABLE DOCUMENTS

The following list contains specifications and publications, parts of which were used in this documentation.

Title	Author
I.431	CCITT
G.703, G.704, G.732, G.733, G.742, G.772, G.823	CCITT
1 TR 231	FTZ
ISDN PRIMARY RATE INTERFACE SPECIFICATION	AT&T
DIGITAL MULTIPLEXED INTERFACE TECHNICAL SPECIFICATION	AT&T
TECHNICAL ADVISORIES	AT&T
ISDN PRI INTERFACE REQUIREMENT SPECIFICATION	B. CARLSON / R. PACK SIEMENS AG - AUT E33 D1
ISDN PRI INTERFACE FUNCTIONAL SPECIFICATION	B. CARLSON / R. PACK SIEMENS AG - AUT E33 D1
ISDN PRI INTERFACE DESIGN SPECIFICATION	B. CARLSON / R. PACK SIEMENS AG - AUT E33 D1
T1/CEPT (PCM-30) PRIMER	ROCKWELL
R8070 T1/CEPT PCM TRANSCEIVER DESIGNER'S GUIDE	ROCKWELL
R8070 T1/CEPT PCM TRANSCEIVER DATA SHEET	ROCKWELL
SMART ANALOG DATA BOOK	CRYSTAL
COURSE FOLDER TELECOM-PRI	SIEMENS AG , TRAINING CENTER FOR MICROELECTRONICS
THE VME-BUS SPECIFICATION	MOTOROLA

KERMIT

User Manual for Protocol Tester K1197

SIEMENS AG

Software Version	:	V2.0
System Version	:	MF 2.2
Revision Date	:	November, 1991

1	INTRODUCTION	1-1
1.1	KERMIT FEATURES	1-2
2	SET-UP	2-1
2.1	PHYSICAL CONFIGURATION	2-1
2.2	LOCAL/REMOTE CONTROL	2-1
2.3	ACCESSING PROCEDURE	2-2
3	KERMIT SCREEN LAYOUT	3-1
3.1	STATUS LINE (LINE 1)	3-1
3.2	DATA DISPLAY (LINES 2 - 17)	3-1
3.3	COMMAND LINES (LINES 18 - 21)	3-1
3.4	FUNCTION KEY LINES (LINES 22 - 23)	3-1
4	FUNCTION KEY ASSIGNMENTS	4-1
4.1	KERMIT MENU	4-2
4.1.1	SETUP MENU	4-3
4.1.2	CONNECT MENU	4-6
4.1.2.1	SEND MENU	4-7
4.1.2.2	RECEIVE MENU	4-8
5	KERMIT STATES	5-1
5.1	INTERPRET STATE	5-1
5.2	CONNECT STATE	5-2
5.3	RECEIVE STATE	5-3
5.3.1	TRANSFER PARAMETER	5-4
5.3.1.1	TRANSFER-TYPE	5-4
5.3.1.2	FILE-TYPE	5-4
5.4	SEND STATE	5-5
5.4.1	TRANSFER PARAMETER	5-6
5.4.1.1	TRANSFER-TYPE	5-6
6	REMOTE KERMIT COMMANDS	6-1
7	ERROR MESSAGES	7-1

1 INTRODUCTION

The "Kermit" file transfer protocol is intended for use in an environment where there may be a diverse mixture of computers from different manufactures. All these systems need have in common is the ability to communicate in ASCII over ordinary serial telecommunication lines. The "Kermit" protocol was originally designed at the Columbia University Center for Computing Activities to meet the needs for a local file transfer between main frames and various microcomputers.

"Kermit" implementations are available for all major operating systems for only a nominal fee. If you are not able to acquire a appropriate "Kermit" software, please contact:

KERMIT Distributaion
Columbia University
Center for Computing Activities
7th Floor, Watson Laboratory
612 West 115th Street
NEW YORK, N.T. 10025

The "Kermit" protocol is based upon a character-oriented transmission over a serial telecommunication line. The "Kermit" design allows adaptions to the restrictions and peculiarities of the medium and the requirements of diverse operating systems (baud-rate, parity, etc.). The protocol is carried out by "Kermit" programs on each end of the connection by sending packets back and forth. The sender sends file names, file contents and control information. The receiver acknowledges explicitly (positively - negatively) each single packet.

The connection is being established by the user. In a typical case of usage, the user runs "Kermit" on a microcomputer, enters terminal emulation, connects to a remote host, logs in (if necessary), runs "Kermit" on the remote host and then issues commands to the remote "Kermit" to start a file transfer, "escapes" back to the local computer and issues corresponding commands to the local "Kermit" to start its side of the file transfer.

Basic "Kermit" provides sequential file transfer only. Microcomputer implementations of "Kermit" are also expected to provide terminal emulation, to facilitate the initial connection. Some optional features exist, including a variety of block check types, a mechanism for passing 8 Bit data trough a 7 Bit communication link, etc.

"Kermit" gives the ordinary user the power to establish a reliable error-free connection between any two computers which can perform the "Kermit" protocol.

1.1 KERMIT FEATURES

Please check the following list to determine the functionality provided by the K1197 "KERMIT".

- Local mode
- Remote mode
- Text-file transfer
- Binary data file transfer
- Sending of wildcard files
- Receiving of wildcard files
- Transfer abortion
- timeout handling
- 8 Bit prefixing
- terminal emulation
- Communication parameters
- Transfer of BREAK

INDEX

A

ABORT, 4-7

B

BIS/CHAR=X, 4-5

C

CNF, 5-4

CONNECT, 4-2

CONTROL=X, 4-5

D

DATA, 5-4

DEBUG=OFF, 4-7

DEBUG=ON, 4-7

DR0/DR1, 4-7

E

EXEC, 5-4

F

FINISH, 4-7

H

HELP, 4-2

K

K_SELECT, 4-4

KERM, 4-4

L

Local, 2-1

M

MENU, 4-2

O

OBJ, 5-4

P

PARITY=X, 4-5

Q

QUIT_HELP, 4-2

QUIT_RECV, 4-9

QUIT_SEND, 4-7

R

RECEIVE, 4-6

Remote, 2-1

REOL=X, 4-5

RSPEED=X, 4-5

RSRC, 5-4

S

SEND, 4-6

SETUP, 4-2

SRC, 5-4

STOP_BITS=X, 4-5

T

T_BREAK, 4-6

T_NUL, 4-6

TBIN, 5-4

TEOL=X, 4-5

transfer-type=ASCII, 5-4

transfer-type=BINARY, 5-4

TSPEED=X, 4-5

TT=ASCII, 4-7

TT=BINARY, 4-7

TYPE=BIN, 4-9

TYPE=CNF, 4-9

TYPE=DAT, 4-9

TYPE=EXE, 4-9

TYPE=OBJ, 4-9

TYPE=RSC, 4-9

TYPE=SRC, 4-9

TYPE=USR, 4-9

W

WD0../WD7, 4-7

U

USR, 5-4

2 SET-UP

2.1 PHYSICAL CONFIGURATION

The "Kermit" file transfer uses only the MODEM-Connector on the auxiliary interface module on the back panel of the K1197.

The "Kermit" protocol requires that the other host is able to send and receive characters. Using 7 or 8 Bit ASCII encoding an EIA RS-232 physical connection.

The pin-configuration for the connecting cable depends on the participating hosts. It is mainly depending on the type of transmission control used by the host to which the K1197 should be connected. Please note that the K1197 in transmissions via the modem connector always acts as a terminal.

Remote = to DCE

If the host plug is TO_DTE (to terminal) use a straight through connected cable, if it is a TO_DCE, then a null modem cable is needed (Pin 2/3 reserved if XOF).

(see K1197 USER MANUAL for the pinout for the Remote Port)

2.2 LOCAL/REMOTE CONTROL

Because the terms local and remote are used in the context of the "Kermit" protocol, it is necessary to distinguish these terms from the normal meaning in a K1197 environment. In the "Kermit" terminology the following definitions are valid.

Local: When two machines are connected, the **local** machine is the one which you interact with directly and which is in control of the terminal. The "local Kermit" is the one that runs on the local machine. A local "Kermit" always communicates over an external device (an assigned TTY line, the K1197 modem port, etc.).

Remote: The **remote** machine is the one on the other side of the connection, which you must interact with "through" the local machine. The "remote Kermit" runs on the remote machine. A remote "Kermit" usually communicates over its own console, controlling terminal or standard I/O device. (An exception is the K1197 solution which communicates as a remote "Kermit" via the modem port either)

As pointed out, the modem connector is permanently used by the "Kermit" file transfer application. Remote control in a way performed by K1197 applications cannot be done while "Kermit" is active.

2.3 ACCESSING PROCEDURE

- Step 1 Make sure that the K1197 is connected to the equipment under test using the cables and connectors.
- Step 2 Follow the directions for "SET UP" in the K1197 User Manual.
- Step 3 Load KERMIT using the APPLICATION SELECTION menu:
 Select INTERFACE,
 SIGNALLING APPLICATION,
 KERMIT
- Step 4 After loading, switch to KERMIT using the ENTER APPLICATION menu.

The package is ready to use after loading is completed.

When the software has been stopped with the functions FINISH or ABORT, it can be restarted by entering KERMIT.

3 KERMIT SCREEN LAYOUT

The screen layout consists of four groups of lines. Line 1 is a status line. Lines 2 to 17 are reserved for data display. Lines 18 to 21 are command lines, and lines 22 and 23 are function key lines.

3.1 STATUS LINE (LINE 1)

The format of line 1 is:

CPU1	DR0	KERMIT	FILE-TRANSFER	1988-06-08	14:33:51
------	-----	--------	---------------	------------	----------

Starting from left to right, the first field shows the current disk drive. DR0 is the floppy disk drive on the left and DR1 is the disk drive on the right. The second field is the "Kermit" id-string. The last field is the real time clock with date and time shown.

3.2 DATA DISPLAY (LINES 2 - 17)

Line 2 to 17 are used for reporting information about the file transfer. The report format is illustrated later on. This group of lines is also used for displaying help information and selecting configuration parameters of the connection.

3.3 COMMAND LINES (LINES 18 - 21)

Lines 18-21 are command lines showing the commands being entered by the user, using the attached keyboard and the resulting command output.

3.4 FUNCTION KEY LINES (LINES 22 - 23)

Lines 22 and 23 displaying the current function key menu. The various function key menus included in the "Kermit" file transfer are described later on. Line 22 displays the functions that are accessed by the SHIFT f1-f8 key combination and line 23 displays the functions that are accessible by the f1-f8 keys.

4 FUNCTION KEY ASSIGNMENTS

The function key menus are organized in a tree structure with KERMIT being the top of the tree. Entry to a lower level is done by the appropriate function key. Return to the higher level is done with the shift f1 key.

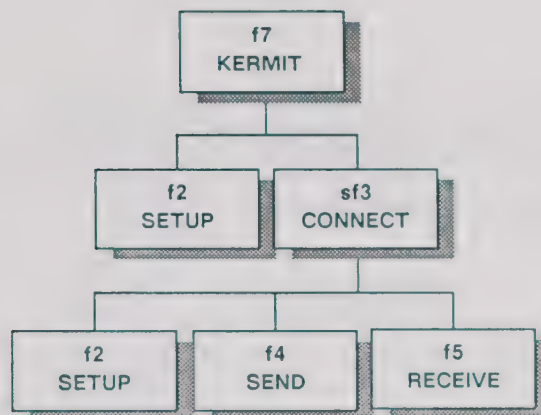


Figure 4-1

4.1 KERMIT MENU

When the "Kermit" package was successfully loaded, the following menu is displayed. This menu allows the user to select the state of the "Kermit" application. The user can either stay at this menu (the "Kermit" here acts as a remote "Kermit") or change to the connect state (in which the "Kermit" acts as a local "Kermit"). It also provides access to the setup function key menu.



1	MENU	2		3	CONNECT	4		5		6		7		8	
1	HELP	2	SETUP	3		4		5		6		7		8	

KERMIT MENU: Displays "KERMIT" functions



HELP: Explains "KERMIT" functions

MENU	: Stop KERMIT
CONNECT	: Changes "Kermit" state to connect and go to connect function key menu.
HELP / QUIT_HELP	: Displays / quits the description of the current function key menu.
SETUP	: Go to the setup function key menu.

4.1.1 SETUP MENU

The setup function key menu allows the selection of connection parameters by using function and cursor keys. All available parameter-settings are displayed on the screen. The current configuration is highlighted in blue. The following parameters are defined:



SETUP MENU: Displays SETUP functions

DR1		KERMIT FILE-TRANSFER		10.06.1991 10:27:30			
RSPEED=75	TSPEED=75	CONTROL=OFF		BITS/CHAR=5			
RSPEED=300	TSPEED=300	CONTROL=XOF		BITS/CHAR=6			
RSPEED=1200	TSPEED=1200	CONTROL=DTR		BITS/CHAR=7			
RSPEED=2400	TSPEED=2400	CONTROL=DTR&XOF		BITS/CHAR=8			
RSPEED=3600	TSPEED=3600						
RSPEED=4800	TSPEED=4800						
RSPEED=7200	TSPEED=7200	PARITY=NONE		REOL=CR			
RSPEED=9600	TSPEED=9600	PARITY=ODD		REOL=LF			
RSPEED=14400	TSPEED=14400	PARITY=EVEN		REOL=CRLF			
RSPEED=16000	TSPEED=16000	STOP BITS=1.0		TEOL=CR			
RSPEED=19200	TSPEED=19200	STOP BITS=1.5		TEOL=LF			
		STOP BITS=2.0		TEOL=CRLF			
KERMIT Version 2.0 Rev. 0							
KERM>							
1 KERM	2	3	4	5	6	7	8
1 HELP	2	3	4 H_SELCT	5	6	7	8



HELP: Explains SETUP functions

KERM	: Returns to the "Kermit" function key menu.
HELP / QUIT_HELP	: Displays / quits the description of the current function key menu.
K_SELECT	: Select configuration parameters

Use the arrow keys to position cursor to desired selection
Current printer configuration is highlighted in blue

RSPEED=X	(--) Receiving speed at the modem connector. Where X has to be one of the following values: 75, 300, 1200, 2400, 3600, 4800, 7200, 9600, 14400, 16000, 19200.
TSPEED=X	(--) Transmission speed at the modem connector. Where X has to be one of the following values: 75, 300, 1200, 2400, 3600, 4800, 7200, 9600, 14400, 16000, 19200.
CONTROL=X	(--) Mode of transmission control. Where X has to be one of the following: OFF, XOF, DTR, DTR&XOF.
PARITY=X	(--) Used parity for transmission. Where X has to be one of the following: NONE, ODD, EVEN.
STOP_BITS=X	(--) Number of stop bits in transmission. Where X has to be one of the following values: 1.0, 1.5, 2.0.
BITS/CHAR=X	(--) Number of bits per character. Where X has to be one of the following values: 5, 7, 8.
REOL=X	(--) End of line sequence which indicates the end of a logical line from the remote machine. Where X has to be one of the following: CR, LF, CRLF.
TEOL=X	(--) End of line sequence which will be transmitted to the remote machine to terminate a logical line. Where X has to be one of the following: CR, LF, CRLF.

4.1.2 CONNECT MENU

The connect function key menu is displayed while "Kermit" is in the connect state. The connect state is entered when the K1197 "Kermit" is wanted as a local "Kermit". It also provides access to the setup, send, and receive function key menus.

The function keys are explained in the following:



1 KERM	2	3	4 T_BREAK	5 T_NUL	6	7	8
1 HELP	2 SETUP	3	4 SEND	5 RECEIVE	6	7	8

CONNECT MENU: Displays CONNECT functions



HELP: Explains CONNECT functions

KERM	: Returns to the "Kermit" function key menu.
T_BREAK	: Transmits a NUL (0x00) for the period of 300 ms.
T_NUL	: Transmits a single NUL (0x00).
HELP / QUIT_HELP	: Displays / quits the description of the current function key menu.
SETUP	: Go to the setup function key menu.
SEND	: Go to the send function key menu.
RECEIVE	: Go to the receive function key menu.

4.1.2.1 SEND MENU

The send function key menu enables the user to run the local send-part of the "Kermit" file transfer protocol. It should be entered **after** the user has executed a receive command on the **remote** host. The user can select the type of transmission, i.e. if he wants to transfer files with ASCII or non ASCII coded information.

The function keys have the following meaning:



SEND MENU: Displays SEND functions

1	QUIT_SEND	2	DR1	3	FINISH	4	ABORT	5		6		7		8	
1	HELP	2		3		4		5	FF=ASCII	6		7		8	DEBUG=OFF



HELP: Explains SEND functions

QUIT_SEND	: Returns to the connect function key menu.
DR0 / DR1 / WD0 / .. / WD7	: Select disk drive 0 / select disk drive 1 / select winchester partition 0 / .. / partition 7
FINISH	: Terminates the current "Kermit" session and the "Kermit" program. The local "Kermit" sends an error packet to the remote "Kermit" and returns to the main menu. Enter KERMIT to restart the program.
ABORT	: Terminates the current "Kermit" session and the "Kermit" program. The local "Kermit" returns to the main menu. Enter KERMIT to restart the program.
HELP / QUIT_HELP	: Displays / quits the description of the current function key menu.
TT=ASCII / TT=BINARY	: Switch to ASCII / BINARY transfer mode. In ASCII mode the transferred file is to be expected in compressed K1197 editor file format and should not contain any non standard ASCII characters. The transmission is done in a manner that the remote "Kermit" is able to create a file readable by an editor. / In binary mode the file is transferred bit transparently to the remote "Kermit". Thus there is no restriction of file contents.
DEBUG=OFF / DEBUG=ON	: Switch debug mode ON/OFF. In DEBUG=OFF mode no additional information is displayed on the screen. DEBUG=ON mode displays all received characters from the other "Kermit" on the screen. There will be no interpretation of received data.

4.1.2.2 RECEIVE MENU

The receive function key menu enables the user to run the local receive-part of the "Kermit" file transfer protocol. It should be entered **after** the user has executed a send command on the **remote** host. In connected mode the user can select the type of transmission, i.e. if he wants to transfer files with ASCII or non ASCII coded information.

The function keys have the following meaning:



RECEIVE MENU: Displays RECEIVE functions

1	QUIT_RECV	2	DR1	3	FINISH	4	ABORT	5		6		7		8	
1	HELP	2		3		4		5	FF-ASCII	6	TYPE=SRC	7		8	DEBUG=OFF



HELP: Explains RECEIVE functions

QUIT_RECV	: Returns to the connect function key menu.
DR0 / DR1 / WD0 / .. / WD7	: Select disk drive 0 / select disk drive 1 / select winchester partition 0 / .. / partition 7
FINISH	: Terminates the current "Kermit" session and the "Kermit" program. The local "Kermit" sends an error packet to the remote "Kermit" and returns to the main menu. Enter KERMIT to restart the program.
ABORT	: Terminates the current "Kermit" session and the "Kermit" program. The local "Kermit" returns to the main menu. Enter KERMIT to restart the program.
HELP / QUIT_HELP	: Displays / quits the description of the current function key menu.
FT=ASCII / FT=BINARY	: Switch to ASCII / BINARY transfer mode. In ASCII mode the transfered file will be stored in compressed K1197 editor file format and should not contain any non standard ASCII characters. The transmission is done in a manner that the remote "Kermit" is able to create a file readable by an editor. / In binary mode the file is transfered bit transparently to the remote "Kermit". Thus there is no restriction of file contents.
TYPE=SRC / TYPE=USR / TYPE=CNF / TYPE=RSC / TYPE=EXE / TYPE=BIN / TYPE=DAT /TYPE=OBJ	: Specifies the file type designator with which the received file will be created.
DEBUG=OFF / DEBUG=ON	: Switch debug mode ON/OFF. In DEBUG=OFF mode no additional information is displayed on the screen. DEBUG=ON mode displays all received characters from the other "Kermit" on the screen. There will be no interpretation of received data.

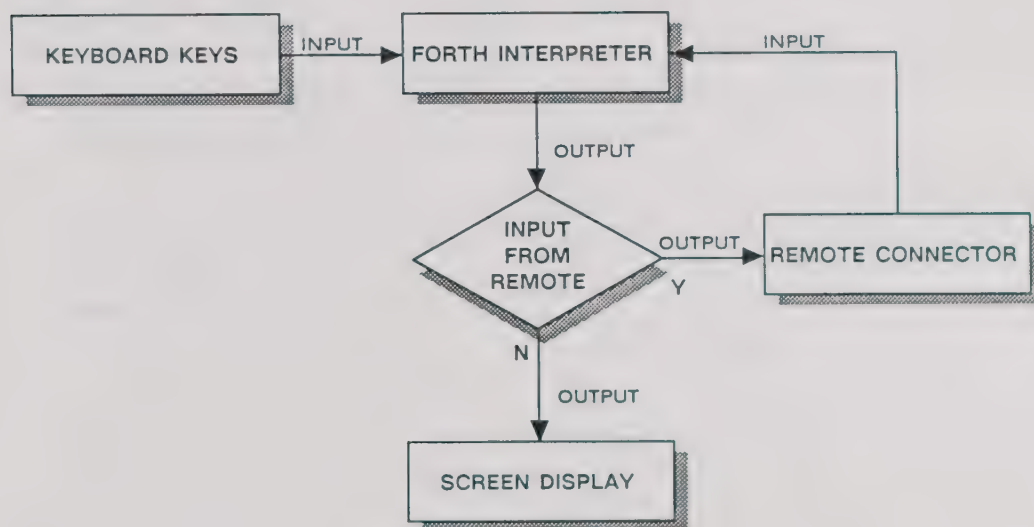
5 KERMIT STATES

The "Kermit" file transfer has two main states INTERPRET and CONNECT, representing the local and remote "Kermit" respectively. On initial startup, the "Kermit" file transfer is in INTERPRET state. The SEND and RECEIVE states are substates of both INTERPRET and CONNECT state.

5.1 INTERPRET STATE

When the "Kermit" file transfer is in the INTERPRET state it interprets and executes all input via the modem connector as FORTH commands. The command execution has the same effects as if the command has been entered on the keyboard. The screen output of these commands will be sent to the modem port.

However, there are some FORTH words which will produce their outputs in the display RAM only. "Kermit" provides a list of words which are enabling the other "Kermit" to reconfigure the physical connection and to activate the SEND and RECEIVE file transfer protocol parts. (See chapter 6). Normal keyboard input appears in the command window and is interpreted as well.



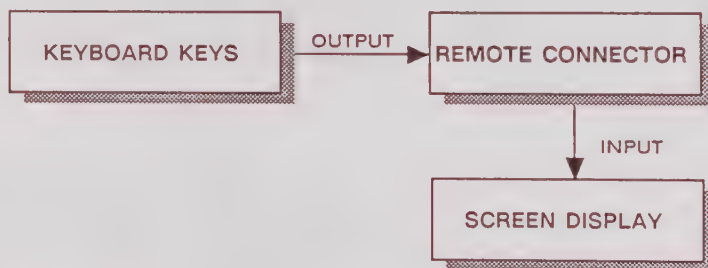
5.2 CONNECT STATE

The CONNECT state is used to set up a connection with the remote host. The local keyboard input is sent character by character via the modem connector to the remote host. The local user may login to the remote host (if neccessary) and start the remote "Kermit". Prerequisite of the correct interface configuration is required. The user may use the setup function key menu to do this. The local user may also execute other, non "Kermit" dependent commands. All incoming characters will be displayed on the screen.

No FORTH command input via the keyboard possible in this state. The remote host is expected to echo all keyboard inputs since there is no local echo active.

Note that no standard terminal emulation is done when displaying the received characters on the screen. As a consequence control character sequences remain uninterpreted.

However, the setup function key menu contains two control character definitions, TEOL and REOL. The parameter TEOL specifies the control characters which are transmitted when the keyboard key CR is hit. The parameter REOL specifies the control charactersto be received to generate a local CR on the screen.



5.3 RECEIVE STATE

When "Kermit" is in RECEIVE substate it performs the "Kermit" file transfer protocol receive part. Data received from the connector modem is collected and written on the selected disk. It is expected that the corresponding "Kermit" file transfer protocol send part is running on the other host. While the "Kermit" is in RECEIVE state transfer information is being reported on the screen.

There are two ways to enter the RECEIVE state:
If "Kermit" acts as a local "Kermit" and the receive function key menu is entered, a prompt asking the user for a filename is displayed on the command window. As soon as the user hits CR, the "Kermit" is in RECEIVE state. If the user has specified a filename, the received file is stored with this name. In the case that only CR is entered, the file will have the same name as on the remote host. It is also possible to receive a group of files without entering the RECEIVE state repeatedly. Once the RECEIVE state is entered, a change of the transfer parameters (transfer-type, file-type) has no effect.
To enter the RECEIVE state as a remote "Kermit" a appropriate word is defined with filename, transfer-type, and file-type as parameters (see chapter 6). After reception of this command in the INTERPRET state the "Kermit" goes immediately into RECEIVE state.
After both "Kermits" have successfully established a connection some information (shown below) is displayed on the screen.
The screen layout is the following:

DR1		KERMIT FILE-TRANSFER				10.06.1991 10:27:30									
FILENAME															
No. of FRAMES		:	Quantity of files :												
No. of Retries		:													
ERRORSTATE		:													
Bytes received		:	whole Bytes received :												
FILE1		FILE2		FILE3		FILE4		FILE5							
				TRANSFER COMPLETED											
KERM>															
RECV-FILENAME															
1	QUIT_RECV	2	DR1	3	FINISH	4	ABORT	5		6		7		8	
1	HELP	2		3		4		5	TT=ASCII	6	TYPE=SFC	7		8	DEBUG=OFF

5.3.1 TRANSFER PARAMETER

5.3.1.1 TRANSFER-TYPE

The transfer-type specifies in which way the file is to be generated.

transfer-type=ASCII indicates, that the transfered file contains only ASCII coded character. The file will be stored in the K1197 compressed editor file format.

transfer-type=BINARY indicates, that no change of any received data will be done. The transfer is strictly bit transparent.

5.3.1.2 FILE-TYPE

The file-type specifies the internal K1197 file type designator which is used at the creation of the file. The meaning of the file-type is application dependent. Following File-Types are used normally:

SRC	Files created with the Editor
USR	User Files
CNF	Conformance Files
RSRC	Remote Source Files
EXEC	Operating System Binary Files
TBIN	Test Manager Binary Files (TOBJ)
DATA	Data recording Files
OBJ	ALOAD Files and FORTH Binary Files

The file type can be displayed by using the command **DIRL** (Long Directory).
(see USER MANUAL Section 3)

5.4 SEND STATE

While "Kermit" is in the SEND substate it performs the "Kermit" file transfer protocol send part. Data from the selected disk is read and send via the connector modem. It is expected that a corresponding "Kermit" file transfer protocol receive part is running on the other host. In this state some state transfer information is being reported on the screen.

There are two ways to enter the SEND state.

If "Kermit" acts as a local "Kermit" and the send function key menu is entered, a prompt asking the user for a filename is displayed on the command window. As soon as the user hits CR, the "Kermit" is in SEND state. All files matching the entered file specification will be send to the other host. In this way it is also possible to send a group of files by only one command. Once the SEND state is entered, a change of the transfer parameters (transfer-type, file-type) has no effect.

To enter the SEND state as a remote "Kermit" an appropriate word is defined with filename and transfer-type as parameters (see chapter 6). After reception of this command in the INTERPRET state the "Kermit" goes immediately into SEND state.

After both "Kermits" have successfully established a connection some information (shown below) is displayed on the screen.

The screen layout is the following:

DR1		KERMIT FILE-TRANSFER				10.06.1991 10:27:30			
FILENAME No. of FRAMES : No. of Retries : ERRORSTATE : Bytes transfered : Quantity of files : whole Bytes transfered :									
FILE1		FILE2		FILE3		FILE4		FILE5	
					TRANSFER COMPLETED				
KERM>									
SEND-FILENAME									
1	QUIT_SEND	2	DR1	3	FINISH	4	ABORT	5	
6		7		8					
1	HELP	2		3		4		5	TT=ASCII
6	TYPE=SRC	7		8	DEBUG=OFF				

5.4.1 TRANSFER PARAMETER

5.4.1.1 TRANSFER-TYPE

The transfer-type specifies in which way the file is to be generated.

transfer-type=ASCII indicates, that the transfered file contains only ASCII coded character. The file will is to be expected in the compressed K1197 editor file format.

transfer-type=BINARY indicates, that no change of any received data will be done. The transfer is strictly bit transparent.

6 REMOTE KERMIT COMMANDS

The following commands are defined for using the K1197 as a remote "Kermit".

- SEND filename (binary-transfer)
- RECEIVE (filename) (binary-transfer) (file-type)
- DIRECTORY
- SET config-parameter

Where:

filename: Standard K1197 file specification (see K1197 USER MANUAL, section 4.12.4)
 binary-transfer: If transfer-type=BINARY should be used for transmission, then specify single character 'B'. Transfer-type=ASCII is used if nothing is specified. (See 5.3.1.1 and 5.4.1.1, respectively) .

file-type: Selects the internal file type designator which is used at the file creation. Following file-types are possible:

'TYPE=SRC'
 'TYPE=USR'
 'TYPE=CNF'
 'TYPE=RSC'
 'TYPE=EXE'
 'TYPE=BIN'
 'TYPE=DAT'
 'TYPE=OBJ'

The file type designator SRC is used by default.

config-parameter: Modifies the current configuration of the modem connector. The following parameters are possible:

'RSPEED=75'	'CONTROL=OFF'
'RSPEED=300'	'CONTROL=XOF'
'RSPEED=1200'	'CONTROL=DTR'
'RSPEED=2400'	'CONTROL=DTR&XOF'
'RSPEED=3600'	'PARITY=NONE'
'RSPEED=4800'	'PARITY=ODD'
'RSPEED=7200'	'PARITY=EVEN'
'RSPEED=9600'	'STOP_BITS=1.0'
'RSPEED=14400'	'STOP_BITS=1.5'
'RSPEED=16000'	'STOP_BITS=2.0'
'RSPEED=19200'	'BITS/CHAR=5'
'TSPEED=75'	'BITS/CHAR=6'
'TSPEED=300'	'BITS/CHAR=7'
'TSPEED=1200'	'BITS/CHAR=8'
'TSPEED=2400'	
'TSPEED=3600'	
'TSPEED=4800'	
'TSPEED=7200'	
'TSPEED=9600'	
'TSPEED=14400'	
'TSPEED=16000'	
'TSPEED=19200'	

7 ERROR MESSAGES

When entering error states the "Kermit" file transfer displays an error message in the data display. The message is also included in an error packet and send to the other "Kermit". The error messages are standard ASCII coded. K1197 error conditions caused by the local file system result in the display of an error message and a negative integer. The meaning of error codes is as follows:

- 1 Invalid disk command
- 2 Error in command argument(s) to disk I/O function
- 3 File does not exist
- 4 Unrecoverable physical I/O error
- 5 Nonexistent I/O device
- 6 Bad file descriptor
- 7 File access prohibited
- 8 File already exists
- 9 Error in disk I/O function
- 10 File name error
- 11 End of file
- 12 No space for file system
- 13 Too many files opened
- 14 File system busy
- 15 Floppy disk not loaded
- 16 No 'RTE' file system
- 17 Disk is write protected
- 18 Can't copy file to itself
- 19 Bad source device
- 20 Bad destination device
- 21 No match for wildcard
- 22 Difference in compared files
- 23 File system is write protected

Hints:

If you get trouble with the host in ASCII file-type, try bit transparent.

For Information about "Kermit" connections to SIEMENS Host Equipment see report:

SIEMENS Kommunikation "KERMIT"

Dok.Nr. 6BB4967-0AX00-0B

SPARE PARTS LIST / ERSATZTEILLISTE

Protocol Tester K1197

SIEMENS AG

Revision Date : Januar , 1990

Benennung	Parts Name	Bestellnummer (Order Number)
Gerätesicherung	Main Fuse	D41662-T3150
Batterie	Back-up Battery	W73084-L1001-B1
Gehäusedeckel	Cover-Top	C73249-A6-D31
Gehäuseboden	Cover-Bottom	C73249-A2-C56
Gehäusefront	Front Bezel	C73249-A6-D33
Gehäuserückseite	Back Bezel	C73249-A2-C69
Tragegriff m. Blenden	Handle + Cover Plates	C73249-A6-D32
Gehäuseseitenfuß	Side Foot	C73249-A6-C8
Gehäuseaufstellfuß	Standoff Foot	C73249-A6-D35
Netzschalter	Main Switch	V42262-Z54-A2
Netzteil	Power Supply	W73048-A3003
40MB Festplattenlaufwerk	40MB Winchester Drive	W73089-G12
Floppy-Disk Laufwerk	Floppy Disk Drive	W73089-G7
Monitor	Monitor	C73249-A6-B30
Tastatur	Keyboard	S44034-K2091-A704
Tastatur-Fußgummi m. Beschriftung	Keyboard Standoff Rubber	C73249-A6-D36
Tastaturkabel	Keyboard Cabel	V45591-A100-A1
Lüfter	Fan	W73087-A1-A80
Keyboard Connector Bgr. m. RESET-Taste	Keyboard Connector	S44035-K5528-A801
Aux. Interface Bgr.	Aux. Interface	S44035-K5527-A801
Grundverdrahtung	Back Plane	S44035-K5529-A801
Hauptrechner Bgr.	CPU2	C73249-A2-L9
Verbindungsbaugruppe	Crossconnect board	C73249-A2-L8
Verbindungsbaugruppe	Interconnect board	C73249-A2-L6
Send / Receive Bgr.	FECP2	S44035-K5565-B801
ISDN PRI-Baugruppe	PRI	S44035-K5555-B801
PRI Connectormodul CEPT	PRI connector module CEPT	S44035-K5558-A801

PRI Connectormodul T1	PRI connector module T1	S44035-K5562-A801
ISDN BAI2-Baugruppe	BAI2	C73249-A2-L10
BAI2 Connectormodul S ₀	BAI2 connector module S ₀	C73249-A2-B20
BAI2 Connectormodul U _{po}	BAI2 connector module U _{po}	C73249-A2-B21

)

)

)

